Il pensiero compresso

Una fenomenologia del minimo descrittivo e della profondità computazionale

---

Proemio: Dove KC non arriva

La complessità di Kolmogorov e l'entropia di Shannon sono strumenti di misura della *sorpresa* e della *ridondanza*. Dicono quanto sia lungo il resoconto più parsimonioso di una stringa, quanto sia comprimibile un messaggio, quanta informazione venga trasmessa su un canale. Sono entrambe misure di economia sintattica: operano sulla superficie di ciò che è scritto, trasmesso, codificato, senza interrogarsi su quanto quel testo abbia dovuto *attraversare* per essere prodotto. È precisamente questo silenzio — la cecità della complessità descrittiva rispetto alla storia computazionale degli oggetti — che apre lo spazio in cui questa monografia intende muoversi.

La tesi centrale non è che il pensiero sia risparmio di bit. È qualcosa di più sottile e, a giudizio di chi scrive, di più fedele all'esperienza effettiva del comprendere: che il pensiero sia la ricerca di oggetti *profondi* nel senso tecnico che Bennett ha dato a questo termine — rappresentazioni che siano insieme comprimibili e costose da generare, che portino inscritta nella propria struttura una storia computazionale irriducibile. Capire non è trovare la descrizione più corta; è trovare il *programma giusto*, quello che gira con efficienza e che, a partire da pochi simboli, dispiega un'estensione arbitrariamente ricca di conseguenze. Il fallimento di questa distinzione genera confusioni tanto nella teoria della cognizione quanto nelle interpretazioni correnti dei modelli di linguaggio su larga scala.

Il percorso si articola in quattro movimenti. Il primo esamina la struttura formale del triangolo KC–Kt–D e le ragioni per cui nessuna delle tre misure, presa isolatamente, è sufficiente a modellare l'atto del comprendere. Il secondo traduce questo apparato nel dominio della cognizione, con particolare attenzione alla fenomenologia dell'intuizione come transizione di fase nello spazio delle descrizioni. Il terzo analizza il tipo specifico di compressione operata dai modelli neurali — umani e artificiali — e le asimmetrie strutturali tra i due tipi di processo. Il quarto affronta il limite: i domini genuinamente incomprimibili, e cosa accade alla cognizione quando li incontra.

Capitolo I — La profondità come categoria formale

I.1 Il vuoto nella misura

Si consideri una stringa casuale x di lunghezza n, generata da una sorgente bernoulliana equidistribuita. Essa ha complessità di Kolmogorov $K(x) \approx n$: è incomprimibile, nessun programma più corto di x può produrla. Ora si consideri la stringa costante 0^n di uguale lunghezza: $K(0^n) = O(\log n)$, poiché basta un programma del tipo "stampa 0 per n volte". Le due stringhe occupano gli estremi opposti dello spettro di KC. Eppure, a un osservatore che cercasse in esse struttura, profondità, interesse cognitivo, entrambe deluderebbero nella stessa misura: la prima per eccesso di rumore, la seconda per eccesso di banalità.

Questo è il primo segnale che KC non è una misura di *ricchezza strutturale*, bensì di imprevedibilità. Le stringhe cognitivamente interessanti — la sequenza dei decimali di π troncata al milionesimo posto, la struttura di un frattale di Mandelbrot, la partitura di una fuga di Bach, il testo di un teorema arduo — non si collocano né al polo del caos né a quello dell'ordine banale. Occupano una regione intermedia che KC non riesce a distinguere dai rispettivi vicini: un testo ricco di struttura e un testo caotico possono avere KC simile, pur essendo radicalmente eterogenei per ciò che si chiama comunemente profondità.

Charles Bennett comprese che la grandezza mancante era il *tempo di calcolo necessario al programma ottimale* per generare la stringa in questione. Un oggetto è profondo se il suo programma più conciso richiede molto tempo per essere eseguito; è superficiale se può essere generato rapidamente, indipendentemente dal fatto che la sua descrizione sia lunga o corta.

I.2 La profondità logica di Bennett

Si fissi una macchina di Turing universale U. Si indichi con t(p) il numero di passi che U compie eseguendo il programma p prima di arrestarsi. La *profondità logica* di una stringa x al livello di significatività s è:

$$D_s(x) = \min\{\, t(p) : U(p) = x,\ |p| \leq K(x) + s \,\}$$

In prosa: $D_s(x)$ è il tempo minimo di esecuzione tra tutti i programmi che generano x e la cui lunghezza eccede la lunghezza ottimale di al più s bit. Il parametro s controlla la tolleranza: valori piccoli di s costringono a considerare solo i programmi quasi-ottimali in lunghezza; valori più grandi ammettono programmi subottimali che potrebbero tuttavia girare più velocemente. La dipendenza da s è lenta — logarítmica nelle applicazioni tipiche — e non altera la classificazione qualitativa degli oggetti.

Tre classi emergono naturalmente da questa misura.

Le stringhe *banali* hanno sia KC bassa che D bassa: prodotte da programmi brevi che terminano in pochi passi. La stringa costante 0^n è il prototipo: il programma "stampa 0 per n volte" è conciso e veloce.

Le stringhe *casuali* hanno KC alta e D bassa. Se nessun programma breve esiste, il programma più parsimonioso è essenzialmente "copia la stringa" e gira in tempo lineare nella lunghezza. L'incomprimibilità implica che non esista alcun programma breve che richieda tempo esponenziale per arrestarsi. Il caos, paradossalmente, è superficiale.

Le stringhe *profonde* hanno KC bassa e D alta: esiste un programma conciso che le genera, ma quel programma richiede un tempo di esecuzione molto elevato. I decimali di π ne sono il caso paradigmatico: il programma che li produce è breve (un algoritmo per la serie di Machin o per l'algoritmo BBP occupa qualche decina di righe), ma per produrre il milionesimo decimale deve eseguire un numero di operazioni che cresce in modo sostanziale con la precisione richiesta.

I.3 La relazione formale con KC e la sua asimmetria

La profondità logica e la complessità di Kolmogorov sono legate da una relazione di dipendenza asimmetrica. Oggetti con KC bassa possono avere D arbitrariamente alta: programmi concisi possono richiedere tempi di esecuzione arbitrariamente lunghi. Ma oggetti con KC alta non possono avere D alta nel senso profondo: se nessun programma conciso esiste, il programma migliore è quasi identico alla stringa stessa e lo si esegue in tempo lineare.

Più precisamente, per qualsiasi stringa x vale la maggiorazione:

$$D_s(x) \leq 2^{(K(x) + s)}$$

poiché il programma quasi-ottimale deve arrestarsi entro un numero di passi al più esponenziale nella propria lunghezza — altrimenti non si arresterebbe mai. Questa è una stima grossolana ma significativa: la profondità è *al più* esponenziale nella complessità di Kolmogorov. Non esiste un limite inferiore generale: oggetti a bassa KC possono raggiungere la profondità massima.

Bennett ha dimostrato che una stringa casuale ($KC \approx n$) ha profondità al più polinomiale in n. La ragione è che, se nessun programma conciso esiste, il programma più corto è lungo quanto la stringa stessa, e un siffatto programma esegue essenzialmente la copia della stringa in tempo $O(n)$. Il caos è dunque computazionalmente *pianeggiante*, privo di quella storia sedimentata che caratterizza gli oggetti profondi.

Questa asimmetria ha un nome nella letteratura: le stringhe profonde sono dette computazionalmente ricche, nel senso che la loro struttura non è accessibile senza ripercorrere almeno in parte il processo che le ha generate. Il termine non è

metaforico: misura, in unità di passi macchina, quanta storia causale sia inscritta in un oggetto.

I.4 La profondità come misura di storia sedimentata

Il punto filosoficamente più rilevante è che la profondità logica è una misura della *storia computazionale* di un oggetto — di quanta elaborazione irriducibile abbia dovuto aver luogo perché quell'oggetto esistesse nella forma in cui esiste. Un cristallo complesso ha alta profondità perché la sua struttura è il sedimento di un lungo processo di minimizzazione dell'energia libera; un genoma articolato porta inscritta l'intera storia evolutiva della specie che lo ospita. Un teorema difficile ha alta profondità perché la sua dimostrazione — il programma che lo "genera" in modo verificabile — richiede una catena di inferenze lunga e non abbreviabile senza perdita.

Questo trasforma la profondità logica da misura tecnica a categoria interpretativa. La domanda non è più "quanto è lunga la descrizione più corta?" bensì "quanta elaborazione è necessaria per ricostruire questo oggetto a partire dalla sua descrizione più corta?". La risposta discrimina il banale dal caotico, il caotico dal ricco, il ricco dal profondo.

Per la cognizione la conseguenza è immediata e non triviale: capire qualcosa non significa trovarne la descrizione più breve. Significa trovare la descrizione più breve *che sia anche computazionalmente accessibile* — un programma conciso che si possa effettivamente eseguire, che produca previsioni, che si estenda a casi non ancora incontrati. È questa tensione che la complessità di Levin — il secondo vertice del triangolo — formalizza con precisione.

I.5 La complessità Kt di Levin: il costo della velocità

Leonid Levin introdusse negli anni Settanta una misura che bilancia lunghezza del programma e tempo di esecuzione in un'unica grandezza scalare. La complessità Kt di una stringa x è:

$$Kt(x) = \min\{\, |p| + \log_2 t(p) : U(p) = x \,\}$$

Il minimo è preso su tutti i programmi p che producono x. Si sommano la lunghezza del programma (in bit) e il logaritmo in base 2 del suo tempo di esecuzione (in passi macchina). L'uso del logaritmo non è arbitrario: corrisponde al numero di bit necessari per *specificare il tempo* come quantità, e assicura l'omogeneità dimensionale della somma — entrambi gli addendi si misurano in bit.

Il trade-off catturato dalla formula è preciso. Un programma molto breve che richiede tempo esponenziale può avere Kt altrettanto elevata di un programma moderatamente lungo che gira in tempo polinomiale. Il minimo è raggiunto dal programma che bilancia ottimamente le due risorse: né così conciso da richiedere tempo astronomico, né così prolisso da aggiungere descrizione superflua.

Formalmente si dimostra che:

$$K(x) \leq Kt(x) \leq K(x) + \log_2 T^*(x) + O(1)$$

dove $T^*(x)$ è il tempo di esecuzione del programma ottimale in lunghezza per x. La disuguaglianza sinistra è immediata: $\log_2 t(p) \geq 0$ per ogni programma che termini in almeno un passo. La disuguaglianza destra si ottiene osservando che il programma ottimale in lunghezza è sempre un candidato per il minimo in Kt, con contributo logaritmico $\log_2 T^*$.

Esiste inoltre un risultato profondo, dovuto a Levin stesso, che collega Kt alla misura di probabilità algorítmica. Si definisce:

$$Km(x) = -\log_2 (\Sigma_\{p : U(p) = x\} 2^\{-|p|\})$$

che aggrega i contributi di tutti i programmi che producono x, pesati esponenzialmente per la lunghezza. Si dimostra che Kt(x) e Km(x) differiscono al più per una costante additiva:

$$|Kt(x) - Km(x)| = O(1)$$

Questo risultato — la *universalità di Kt* — ha conseguenze notevoli: Kt non è soltanto una misura pragmatica del costo computazionale, ma coincide (a meno di costanti dipendenti dalla scelta di U) con la complessità informazionale ottimale quando si pesano *tutte* le possibili spiegazioni di x. È la misura che un agente con risorse cognitive illimitate ma non infinita pazienza userebbe per valutare la plausibilità di una descrizione — e la sua applicabilità. Questo la rende, dei tre vertici del triangolo, la più vicina a ciò che nella pratica cognitiva si chiama comprensione.

I.6 Il triangolo e la mappa dello spazio cognitivo

I tre vertici si dispongono ora in una struttura coerente.

KC misura l'economia sintattica: quanto è corta la descrizione minima, indipendentemente dal costo di elaborarla.

D misura la storia computazionale sedimentata: quanto tempo occorre al programma più conciso per produrre l'oggetto, ovvero quanto processo irriducibile l'oggetto porta inscritto.

Kt misura l'efficienza cognitiva: il costo totale che bilancia concisione e velocità, la misura che massimizza la plausibilità di una descrizione tra tutte quelle computazionalmente praticabili.

I tre vertici definiscono una topologia dello spazio degli oggetti cognitivi che nessun singolo asse coordina può descrivere adeguatamente. La regione di maggiore interesse

— quella che corrisponde a ciò che si chiama comprensione di un dominio complesso — è caratterizzata da KC bassa, D alta e Kt intermedia. In essa si trovano gli oggetti *profondi e comprensibili*: accessibili a un programma conciso, prodotti da un processo lungo e non abbreviabile, utilizzabili con un costo computazionale gestibile.

Un fisico che comprende la meccanica quantistica possiede una descrizione concisa (le equazioni di Schrödinger, il formalismo degli operatori, le regole di selezione) — KC bassa. Ma quella descrizione porta inscritta una storia di revisioni, anomalie risolte, esperimenti mentali, controversie interpretative — D alta. E il fisico sa *usarla* per generare previsioni in tempi ragionevoli — Kt intermedia. Togliere uno qualsiasi dei tre elementi produce qualcosa di diverso dalla comprensione: un formulario vuoto se manca la profondità; un archivio storico inutilizzabile se manca l'efficienza; un'accumulazione indifferenziata se manca la concisione.

Nel capitolo successivo si traduce questo triangolo nel dominio della fenomenologia cognitiva, con un'attenzione particolare al momento in cui la comprensione *accade* — l'insight, l'Aha-Erlebnis — come transizione discontinua nello spazio KC–D–Kt, e alle ragioni per cui questa discontinuità non è un ornamento narrativo ma una conseguenza strutturale della geometria dello spazio delle descrizioni.

Capitolo II — L'insight come transizione di fase

II.1 Il problema fenomenologico

L'esperienza del comprendere ha una caratteristica che ogni teoria cognitiva è tenuta a spiegare e che le spiegazioni gradualistiche — l'apprendimento come accumulo progressivo, come affinamento iterativo della rappresentazione — faticano a rendere conto: la sua *discontinuità*. Chi lavora a lungo su un problema difficile non sperimenta di regola un avanzamento uniforme verso la soluzione; sperimenta periodi prolungati di stagnazione, seguiti da un momento — spesso improvviso, spesso non direttamente correlato allo sforzo conscio immediatamente precedente — in cui la soluzione si impone con una chiarezza quasi sensoriale. Il termine tedesco Aha-Erlebnis, coniato da Karl Bühler e ripreso da Köhler negli esperimenti sugli scimpanzé, designa proprio questa qualità fenomenologica: un'esperienza non di deduzione ma di *riconfigurazione*, in cui l'oggetto del pensiero cambia di aspetto in modo irreversibile.

La psicologia cognitiva ha accumulato evidenza considerevole sulla struttura di questo fenomeno. Gli studi di Metcalfe e Wiebe negli anni Novanta documentarono la differenza tra risoluzione di problemi ordinari — in cui la sensazione di vicinanza alla soluzione cresce monotonicamente — e insight — in cui la curva di "warmth" resta piatta fino all'istante della soluzione e poi sale verticalmente. Ohlsson ha proposto il concetto di *impasse* come stato cognitivo necessario precedente l'insight: la ristrutturazione avviene precisamente perché la rappresentazione corrente ha esaurito il proprio spazio di ricerca senza trovare soluzione.

La domanda che si vuole porre qui è più radicale: si può dare un modello *formale* di questa discontinuità nel senso che la matematica attribuisce al termine, e non la metafora? La risposta, si sosterrà, è affermativa — e il triangolo KC–D–Kt fornisce la geometria necessaria.

II.2 Lo spazio delle descrizioni come varietà

Si consideri l'insieme di tutte le rappresentazioni che un agente cognitivo può formarsi di un dominio o di un problema. Questa non è una metafora: si intende letteralmente l'insieme di tutti i programmi p tali che U(p) produca una rappresentazione che superi un certo criterio di adeguatezza empirica — che generi previsioni verificate entro una soglia di errore fissata. Si chiami questo insieme $\mathcal{R}(D)$.

Ogni elemento di $\mathcal{R}(D)$ ha coordinate naturali nel triangolo: un valore di KC, un valore di D_s e un valore di Kt. Si può dunque immaginare $\mathcal{R}(D)$ come una nuvola di punti in $\mathbb{R}^3$ con coordinate (KC, D, Kt), vincolata alla sottovarietà definita dalle disuguaglianze del capitolo precedente. La posizione attuale dell'agente è determinata dalla rappresentazione p_0 che egli effettivamente adopera, con coordinate ($|p_0|$, $D_s(U(p_0))$, $Kt(U(p_0))$).

L'apprendimento graduale corrisponde a un movimento continuo in questo spazio: il raffinamento di p_0 in p_1, poi in p_2, descrivendo una traiettoria continua che ottimizza progressivamente le coordinate. L'insight corrisponde a qualcosa di radicalmente diverso: un salto discontinuo a una regione dello spazio che non è connessa per continuità alla posizione corrente.

II.3 La geometria dell'impasse

Si introduce la nozione di *bacino di attrazione* di una rappresentazione. Si considerino tutte le rappresentazioni p' ottenibili da p_0 mediante modifiche locali — aggiungere o rimuovere istruzioni, variare parametri, alterare la struttura di controllo in modo puntuale. Queste formano il vicinato locale $V(p_0)$, e il loro insieme definisce il bacino in cui la ricerca graduale può muoversi.

L'impasse cognitiva corrisponde alla condizione:

$$\forall\ p' \in V(p_0) :\ Kt(p') \geq Kt(p_0)$$

Nessuna modifica locale migliora il costo totale. La rappresentazione è un minimo locale di Kt — non globale, ma un punto da cui nessuna discesa gradiente è praticabile senza prima risalire.

Il problema strutturale è che il minimo globale — la rappresentazione ottimale del dominio — può trovarsi in una regione con KC significativamente inferiore, cioè con struttura descrittiva radicalmente diversa. Per raggiungerlo è necessario attraversare zone di Kt *superiore*: rappresentazioni intermedie peggiori di p_0 in termini di costo, ma posizionate sul versante del bacino di attrazione del minimo globale. Si crea così una

barriera nel paesaggio energetico delle rappresentazioni, del tutto analoga alla barriera di attivazione in chimica: il sistema è intrappolato in un minimo locale e deve acquisire una "fluttuazione" sufficiente per superare la barriera e cadere nel bacino del minimo globale. Nella fisica dei sistemi complessi questo fenomeno si chiama *nucleazione*: la transizione di fase non avviene uniformemente, ma attraverso la formazione di un nucleo critico abbastanza distante dalla posizione corrente da trovarsi già nell'altro bacino.

L'insight è la nucleazione cognitiva.

II.4 Una stima della barriera di Kt

Si supponga che p_0 abbia KC = k_0 e che p* abbia KC = k* < k_0. Esiste una sequenza di programmi $p_0 = q_0, q_1, ..., q = p^*$ che collega i due mediante operazioni elementari sull'insieme di istruzioni di U. La barriera di Kt è:

$$\Delta Kt = \min_{\gamma: p_0 \to p^*} \max_{0 \le i \le n} [\, Kt(q_i) - Kt(p_0) \,]$$

il minimo è preso sul percorso ottimale tra p_0 e p*; il massimo interno misura il picco di costo su quel percorso. Questa è la quantità analoga alla barriera di attivazione: il valore massimo di Kt che deve essere raggiunto lungo il percorso meno costoso.

La barriera è in generale non calcolabile (per l'indecidibilità del problema dell'arresto), ma se ne possono ottenere stime qualitative. Se p_0 e p* hanno strutture di controllo radicalmente eterogenee — uno è iterativo, l'altro ricorsivo; uno opera su sequenze, l'altro su grafi — il percorso che li collega deve attraversare rappresentazioni ibride con KC alta e tempo di esecuzione lungo, generando una barriera elevata. Se invece differiscono solo in un parametro strutturale, la barriera è bassa e la transizione si compie con minimo sforzo.

Questa previsione ha una controparte storica verificabile: le rivoluzioni concettuali più traumatiche — quelle che la storia del pensiero ricorda come cambi di paradigma — corrispondono a transizioni tra rappresentazioni con strutture di controllo radicalmente diverse, ossia a salti attraverso barriere di Kt elevate. La transizione dal sistema tolemaico a quello copernicano, il passaggio dalla meccanica classica alla quantistica, lo spostamento dalla genetica mendeliana alla genetica molecolare: in ciascun caso, la nuova rappresentazione non è un raffinamento della precedente ma un cambio di struttura computazionale — il tipo di oggetto su cui si opera (epicicli vs. orbite ellittiche; traiettorie continue vs. stati quantizzati; caratteri fenotipici vs. sequenze nucleotidiche) è radicalmente diverso.

II.5 Il ruolo della profondità nella resistenza al cambiamento

La geometria KC–D–Kt illumina con insolita nitidezza la *resistenza al cambiamento delle rappresentazioni profonde*. Una rappresentazione con alta profondità logica D è, per definizione, costosa da generare: il suo programma ottimale richiede molto tempo per

produrla. Abbandonarla equivale a perdere una storia computazionale sedimentata che non può essere ricostituita rapidamente.

Se l'agente si trova alla rappresentazione p_0 con $D_s(U(p_0))$ molto elevata, la traiettoria verso p^* deve attraversare rappresentazioni provvisorie a bassa profondità — le strutture ibride sulla cresta della barriera. Questo comporta una riduzione temporanea di D: il soggetto abita per un certo tempo rappresentazioni che non hanno ancora sedimentato la storia computazionale della destinazione.

Questa riduzione temporanea corrisponde a ciò che i teorici della creatività chiamano *regressione al servizio dell'Io* (Ernst Kris) o, in termini più neutri, desimbolizzazione provvisoria: il momento in cui la comprensione viene smontata prima di essere riassemblata in una forma nuova. Non è un difetto del processo ma una necessità strutturale: la barriera di Kt richiede di attraversare zone a bassa profondità per raggiungere profondità nuove. Chi non riesce a tollerare questa riduzione temporanea — chi non sopporta di "non capire più" ciò che credeva di capire — rimane intrappolato nel minimo locale.

II.6 Condizione di transizione e ε-accessibilità

Si formalizza la condizione che determina quando una transizione è spontanea. Il *guadagno di Kt* è:

$$\Delta Kt_gain = Kt(p_0) - Kt(p^*) > 0$$

Il percorso $\gamma = (q_0, q_1, ..., q)$ da p_0 a p^* è *ε-accessibile* se:

$$max_\{0 \le i \le n\} \; Kt(q_i) \le Kt(p_0) + \varepsilon$$

per una soglia $\varepsilon > 0$ che riflette la capacità dell'agente di tollerare rappresentazioni provvisoriamente peggiori. La transizione è spontanea se e solo se esiste un percorso ε-accessibile per ε sufficientemente piccolo. Se non esiste alcun siffatto percorso, la transizione richiede una perturbazione esterna: un esempio, una metafora, un'analogia che collochi l'agente su un punto del percorso oltre la cresta della barriera, da cui la discesa verso p^* sia spontanea.

La discontinuità dell'insight è dunque un fatto *topologico*: riflette la disconnessione — nella metrica di Kt — tra i bacini di attrazione delle due rappresentazioni. Non dipende dal livello di intelligenza dell'agente o dalla quantità di informazione a sua disposizione, ma dalla geometria dello spazio delle descrizioni del dominio specifico.

II.7 Gli artefatti cognitivi come riduttori di barriera

Un'estensione rilevante riguarda il ruolo degli artefatti cognitivi — diagrammi, notazioni, strumenti simbolici — nel ridurre ε necessario per rendere accessibile una transizione. La notazione algebrica di Viète e Descartes non ha cambiato la

matematica che descriveva, ma ha reso ε-accessibili percorsi tra rappresentazioni che prima richiedevano un ε così grande da essere di fatto inaccessibili per qualsiasi agente cognitivo reale.

In termini formali, un artefatto cognitivo può essere modellato come una riduzione del costo dei programmi intermedi: lo strumento fornisce subroutine che abbreviano i programmi q_i sul percorso di transizione, riducendo il massimo di Kt sul percorso senza alterare le posizioni di p_0 e p^* nello spazio delle descrizioni. La barriera si abbassa; la transizione diventa praticabile.

Questo offre una prospettiva formale su ciò che Edwin Hutchins chiama *cognizione distribuita*: il sistema cognitivo rilevante non è il cervello isolato, ma il sistema cervello–artefatto–ambiente, e la funzione degli artefatti è abbassare le barriere di Kt che il cervello da solo non potrebbe attraversare. La notazione matematica, la scrittura, il disegno tecnico, i modelli computazionali sono tecnologie di riduzione delle barriere cognitive, e la storia intellettuale è in larga misura la storia di quali transizioni concettuali esse hanno reso per la prima volta ε-accessibili.

Capitolo III — La compressione nei sistemi neurali: umani e artificiali

---

III.1 Due tipi di compressione

Il cervello umano e un modello di linguaggio di grandi dimensioni comprimono entrambi. Questa affermazione, nella sua nudità, è incontestabile: entrambi i sistemi estraggono struttura regolare da corpora di esperienza, costruiscono rappresentazioni che generalizzano oltre i casi incontrati, riducono la varianza dell'input a categorie e relazioni stabili. Il punto su cui la discussione ordinaria si inceppa è che "comprimere" non è un'operazione unica — è una famiglia di processi radicalmente distinti, e la distinzione che conta non è quella tra "vera comprensione" e "mera statistica" (dicotomia filosoficamente naïve e computazionalmente vacua), bensì quella tra *quale tipo di compressione* ciascun sistema realizza, e dove essa si colloca nel triangolo KC–D–Kt.

La tesi di questo capitolo è che i due sistemi realizzano compressioni strutturalmente asimmetriche: i modelli di linguaggio ottimizzano prevalentemente nella direzione di Kt bassa — rappresentazioni efficienti, rapidamente attivabili, distribuite su parametri continui — mentre la cognizione umana costruisce, attraverso processi molto più lenti e costosi, rappresentazioni con D alta, cioè con profondità logica elevata. La differenza non è di grado ma di tipo, e ha conseguenze precise su ciò che ciascun sistema può e non può fare.

III.2 La compressione nei modelli di linguaggio: la geometria degli embedding

Un modello Transformer di grandi dimensioni può essere descritto, a un certo livello di astrazione, come una funzione parametrica:

$$f_\theta : \mathcal{V}^* \to \Delta(\mathcal{V})$$

che mappa sequenze di token dello spazio $\mathcal{V}^*$ nella distribuzione di probabilità sul vocabolario $\mathcal{V}$ del token successivo. L'addestramento minimizza la perdita di cross-entropia sul corpus:

$$\mathcal{L}(\theta) = -\mathbb{E}_{(x_1,\ldots,x) \sim \mathcal{D}} \left[\Sigma_i \log p_\theta(x_i \mid x_1,\ldots,x_{i-1}) \right]$$

Questo obiettivo è, in senso preciso, un obiettivo di compressione: minimizzare la cross-entropia equivale a minimizzare la lunghezza del codice che il modello assegna al corpus, misurata in bit. Per il teorema di Shannon, il codice ottimale assegna a ogni sequenza una lunghezza proporzionale a $-\log_2 p_\theta$ di quella sequenza; minimizzare la perdita è dunque massimizzare la compressione del corpus da parte del modello. Un modello con perdita di cross-entropia H bits/token utilizza H bits per token per "descrivere" il testo — la perdita è, letteralmente, la KC media per token del corpus rispetto al modello.

Questo ha un'implicazione che raramente viene sottolineata: un modello di linguaggio è, per costruzione, un compressore KC nel senso di Shannon, ottimizzato per la lunghezza della descrizione e non per la profondità nel senso di Bennett. La sua funzione obiettivo non contiene alcun termine che premi la generazione di rappresentazioni con alta profondità logica D — che favorisca, cioè, programmi concisi ma computazionalmente costosi. Al contrario, i meccanismi di attention e feedforward che costituiscono il Transformer sono progettati per essere computazionalmente efficienti: l'inferenza deve avvenire in tempo polinomiale nella lunghezza della sequenza, il che vincola strutturalmente il tipo di computazione accessibile.

III.3 La profondità nei Transformer: un limite strutturale

Si consideri più da vicino la struttura computazionale di un Transformer a L strati. Ogni strato applica una combinazione di self-attention e rete feed-forward. L'operazione di attention può essere scritta come:

$$\text{Attention}(Q, K, V) = \text{softmax}\left(QK^T / \sqrt{d_k} \right) \cdot V$$

dove $Q, K, V \in \mathbb{R}^{n \times d}$ sono le matrici di query, chiave e valore, n è la lunghezza della sequenza e d_k è la dimensione delle chiavi. Il costo computazionale di questa operazione è $O(n^2 d)$, e la profondità computazionale dell'intero modello — intesa come numero di operazioni sequenziali non parallelizzabili — è O(L), lineare nel numero di strati.

Questa linearità è cruciale. La profondità computazionale di un Transformer a L strati è L, indipendentemente dalla lunghezza della sequenza e dalla complessità del problema. Ne segue che la profondità logica D delle rappresentazioni che il modello può costruire in una singola passata forward è bounded superiormente da una funzione lineare di L:

$$D(\text{rappresentazione prodotta da Transformer_L}) = O(L \cdot d^2)$$

dove d è la dimensione del modello. Questo non è un difetto contingente correggibile con più parametri: è un vincolo strutturale dell'architettura. Un Transformer, per quanto grande, genera le proprie rappresentazioni attraverso un numero fisso e limitato di passi sequenziali, e la profondità logica delle sue produzioni è limitata da quel numero.

Per confronto, la profondità logica di molti oggetti cognitivamente rilevanti — la dimostrazione di un teorema difficile, la comprensione di un sistema fisico complesso, la pianificazione strategica su orizzonti lunghi — cresce senza limite con la complessità del problema. Non esiste un upper bound polinomiale nella "dimensione" del problema per la profondità logica di una dimostrazione del teorema di Fermat o per la comprensione della meccanica statistica fuori dall'equilibrio.

Il punto non è che i Transformer siano incapaci di trattare questi domini — lo fanno, con efficacia sorprendente — ma che la rappresentazione che essi costruiscono ha necessariamente profondità logica limitata: è una proiezione dello spazio dei problemi profondi su un sottospazio a bassa D, individuato dai parametri del modello addestrato. La qualità di questa proiezione può essere straordinaria, e le sue produzioni possono sembrare — spesso sono — cognitivamente utili. Ma la struttura interna di quella rappresentazione differisce in modo qualitativo da quella che un umano sviluppa dopo anni di pratica in un dominio.

III.4 Chain-of-thought come aumento di profondità

Un'osservazione tecnica qui merita attenzione perché illumina il vincolo appena descritto. La tecnica nota come *chain-of-thought prompting* — l'invito esplicito al modello a "mostrare i passaggi" prima di produrre la risposta finale — aumenta le prestazioni dei Transformer in modo sistematico e riproducibile su compiti che richiedono ragionamento multi-step. Perché?

La risposta, nel quadro KC–D–Kt, è che il chain-of-thought aumenta la profondità computazionale effettiva del processo: invece di produrre la risposta in un'unica passata forward a L strati, il modello esegue una sequenza di passate, ciascuna condizionata sull'output delle precedenti. Se il modello esegue k passate da L strati ciascuna, la profondità computazionale effettiva sale a O(kL), non più costante rispetto alla difficoltà del problema. Le rappresentazioni intermedie prodotte nei passi intermedi fungono da "scratch pad" — una memoria esterna che estende artificialmente la profondità disponibile.

Formalmente, se si indica con p_CoT il "programma" implicito eseguito dal modello con chain-of-thought, si ha:

D_s(output(p_CoT)) > D_s(output(p_direct))

per la stessa query e lo stesso modello. Il chain-of-thought è, in questo senso tecnico, un meccanismo di aumento di profondità: trasforma una computazione limitata in profondità in una sequenza di computazioni limitate, il cui prodotto complessivo può avere profondità superiore alla singola passata.

Questo spiega anche perché il chain-of-thought aiuta su alcuni compiti e non su altri: aiuta precisamente sui compiti la cui soluzione richiede profondità logica superiore al limite strutturale del modello in singola passata, e non migliora — o migliora marginalmente — sui compiti la cui soluzione è già accessibile entro quel limite.

III.5 La compressione umana: induzione di programma e costruzione di profondità

Il sistema cognitivo umano realizza la compressione attraverso meccanismi radicalmente diversi, e la differenza più rilevante non è la velocità di elaborazione né la dimensione della memoria, ma il tipo di oggetti che il processo produce.

L'apprendimento umano, nei domini complessi, non è ottimizzazione di parametri continui su un obiettivo fisso. È, in senso ampio, *induzione di programma*: il sistema costruisce, attraverso l'esperienza, strutture computazionali — schemi, procedure, teorie, modelli causali — che sono esse stesse capaci di generare comportamento. Questa distinzione corrisponde, nel lessico dell'apprendimento automatico, alla differenza tra *model-free* e *model-based learning*: il primo apprende associazioni stimolo-risposta ottimizzate; il secondo costruisce un modello interno del dominio e usa il modello per pianificare.

Ma la distinzione è più profonda di quanto questa dicotomia catturi. I modelli causali che l'apprendimento umano costruisce non sono semplici macchine predittive: sono strutture con alta profondità logica, nel senso che incorporano storia computazionale — catene di inferenze, revisioni, tentativi falliti, analogie — che non è riducibile alla semplice osservazione degli esempi. Un bambino che impara il concetto di numero non apprende una funzione di classificazione che mappa stimoli visivi su etichette numeriche: costruisce una struttura ricorsiva capace di generare la serie dei naturali, di comprendere l'operazione di successore, di estendersi ai negativi e ai razionali. La profondità logica di questa struttura — il tempo computazionale richiesto per generarla a partire da una descrizione minima — è molto superiore a quella di qualsiasi funzione di classificazione addestrata sugli stessi esempi.

III.6 Asimmetria nella direzione dell'ottimizzazione

Si può ora formulare l'asimmetria strutturale in termini precisi. Entrambi i sistemi operano nell'universo del triangolo KC–D–Kt, ma ottimizzano in direzioni diverse.

I modelli di linguaggio di grandi dimensioni minimizzano Kt in modo diretto — il loro obiettivo di training è, come si è mostrato, equivalente a minimizzare la lunghezza descrittiva (KC) delle produzioni, e la loro architettura li vincola a profondità D limitata. La loro posizione nel triangolo è caratterizzata da KC bassa, D bassa (bounded da $O(L \cdot d^2)$), Kt bassa.

La cognizione umana, nei domini profondi, tende verso una regione diversa: KC relativamente bassa (le teorie scientifiche sono comprimibili in equazioni), D alta (la storia computazionale del dominio è inscritta nella comprensione), Kt intermedia o alta (il costo di applicare la comprensione non è trascurabile — un fisico impiega tempo per derivare una previsione dal modello).

L'asimmetria non indica superiorità di un sistema sull'altro in assoluto: indica specializzazione. I modelli di linguaggio eccellono in compiti che richiedono accesso rapido a vaste associazioni con bassa D — produzione linguistica fluente, recupero di fatti, analogie superficiali. La cognizione umana eccelle in compiti che richiedono D alta — pianificazione su orizzonti lunghi, revisione di credenze di fronte a anomalie, costruzione di nuove strutture concettuali in domini senza precedenti.

III.7 Dove le rappresentazioni divergono: un esperimento mentale

Si consideri il seguente esperimento mentale, che rende concreta l'asimmetria. Si presentino a un modello di linguaggio di grandi dimensioni e a un matematico esperto la seguente situazione: un nuovo sistema di assiomi, mai visto prima, con una struttura che non somiglia a nessun sistema classico. Si chieda a entrambi di determinare se una certa proposizione p è dimostrabile nel sistema.

Il modello di linguaggio affronterà il compito attivando pattern associativi che collegano il nuovo sistema agli sistemi noti — troverà somiglianze sintattiche, recupererà dimostrazioni analoghe, produrrà un output plausibile. La qualità dell'output dipenderà dalla prossimità del nuovo sistema ai sistemi incontrati in addestramento: alta per sistemi simili, degradante per sistemi genuinamente nuovi.

Il matematico, per contro, *costruirà* una rappresentazione del sistema: ne esplorerà le conseguenze, identificherà le strutture portanti, formulerà e testerà congetture, svilupperà un'intuizione per le sue proprietà caratteristiche. Il processo è lento — molto più lento dell'inferenza del modello — ma produce una rappresentazione con profondità logica propria: una struttura che porta inscritta la storia della sua costruzione e che può generalizzare in modo affidabile a proposizioni mai considerate.

La differenza non è che il matematico "capisce davvero" in modo mistico. È che la rappresentazione che costruisce ha D alta: incorporate in essa ci sono catene di ragionamento, tentativi falliti, revisioni, analogie elaborate — tutto ciò che KC da sola non cattura. Il modello produce una proiezione di bassa D su uno spazio ad alta D; il matematico costruisce un oggetto che abita quello spazio direttamente.

III.8 Il problema dell'allucinazione come sintomo di bassa profondità

Il fenomeno delle allucinazioni nei modelli di linguaggio — la produzione di affermazioni plausibili ma false — acquista in questo quadro una spiegazione strutturale che va oltre il semplice "il modello non sa quando non sa". L'allucinazione è il sintomo di una proiezione a bassa D che viene scambiata per un oggetto ad alta D.

Quando il modello produce un'affermazione su un dominio che richiede alta profondità — la storia dettagliata di un evento specifico, le proprietà di un sistema fisico in regime non lineare, i dettagli di una dimostrazione matematica — sta operando nel suo regime naturale di bassa D e proiettando il risultato su un dominio che richiede alta D. Il risultato è localmente coerente — la struttura sintattica e la plausibilità locale sono garantite dall'ottimizzazione su KC — ma globalmente inconsistente con la struttura profonda del dominio.

Formalmente: il modello produce una stringa x tale che KC(x) è bassa — x è fluente, coerente localmente, sintaticamente ricca — ma $D_s(x)$ è molto inferiore alla profondità $D_s(x^*)$ della stringa corretta x^*. Il costo di questa produzione in Kt è basso; il costo di verificare la correttezza di x rispetto alla struttura profonda del dominio è invece alto — richiede, appunto, un agente con D alta.

Questa non è una patologia eliminabile con più dati o più parametri: è una conseguenza strutturale del tipo di compressione che il modello realizza. Possono ridurla tecniche che aumentano la profondità computazionale effettiva — chain-of-thought, tool use, accesso a basi di conoscenza verificate — ma non possono eliminarla senza modificare il tipo di compressione realizzata, cioè senza modificare la posizione del sistema nel triangolo KC–D–Kt.

Capitolo IV — I domini incomprimibili e il limite del pensiero come compressione

---

IV.1 Il limite come parte della teoria

Una teoria del pensiero come compressione che non affrontasse i propri limiti sarebbe parziale nel senso deteriore del termine. Il programma di questo capitolo è dunque opposto a quello apologetico: si cercano i domini in cui la compressione fallisce strutturalmente, si analizzano le ragioni formali di quel fallimento, e si trae da esse una conclusione che non è nichilistica ma delimitatrice — una mappa dei confini entro cui la metafora del pensiero come compressione ha potere esplicativo, e oltre i quali richiede supplementi teorici che essa stessa non può fornire.

I limiti che si esaminano sono di tre specie distinte. Il primo è il limite computazionale: i domini per i quali il problema di trovare la descrizione ottimale è indecidibile o

intrattabile — non per incapacità contingente dell'agente, ma per ragioni strutturali legate alla teoria della computazione. Il secondo è il limite epistemico: i domini in cui la compressione è possibile ma la struttura compressa non corrisponde ad alcuna realtà — i casi in cui il mondo è genuinamente casuale, e il pensiero che cerca struttura dove non ne esiste produce illusione piuttosto che conoscenza. Il terzo è il limite fenomenologico: i domini dell'esperienza soggettiva che sembrano resistere strutturalmente a qualsiasi descrizione compressa, non per mancanza di strumenti ma per una ragione che ha a che fare con la natura stessa della prima persona.

IV.2 Il limite computazionale: l'indecidibilità di KC

Il primo e più radicale limite è ben noto nella teoria della computabilità, ma le sue conseguenze cognitive meritano di essere riconsiderate con più attenzione di quanto la trattazione standard conceda.

La complessità di Kolmogorov K(x) di una stringa x non è calcolabile. Non esiste algoritmo che, dato x come input, produca K(x) come output in tempo finito per ogni x. La prova è immediata: se esistesse tale algoritmo, si potrebbe usarlo per risolvere il problema dell'arresto — si potrebbe determinare se un programma p si arresta cercando, tra tutte le stringhe di lunghezza K(x), quella con KC minima che sia output di p, e verificando se p vi converge. Ma il problema dell'arresto è indecidibile per il teorema di Turing, dunque KC non è calcolabile.

Le conseguenze per il pensiero come compressione sono precise. Il programma di trovare la rappresentazione ottimale — nel senso di KC — di un dominio è in generale irrisolvibile: non può essere eseguito da alcun algoritmo in tempo finito. Ciò che la cognizione realizza non è dunque la compressione ottimale, bensì un'approssimazione della compressione ottimale — e la qualità di questa approssimazione dipende dai meccanismi euristici disponibili all'agente.

Questo non invalida il programma teorico: KC rimane lo standard rispetto al quale le approssimazioni vengono misurate, e la profondità logica D rimane una misura sensata anche se non calcolabile in modo esatto. Ma introduce un'asimmetria importante: l'agente può *avvicinarsi* alla compressione ottimale, ma non può sapere con certezza quanto se ne stia avvicinando. La metacognizione — la capacità di sapere quando si capisce davvero — è strutturalmente limitata da questa indecidibilità.

IV.3 La complessità del minimo descrittivo: limiti alla Kt

Levin ha dimostrato un risultato di completezza che è, al tempo stesso, un risultato di ottimismo e di pessimismo. Esiste un algoritmo universale di ricerca — oggi noto come Algoritmo di Levin o *Universal Search* — che, dato un problema con soluzione verificabile in tempo polinomiale, trova la soluzione ottimale in Kt in tempo al più proporzionale a 2^{Kt(x)}, dove x è la soluzione. Formalmente:

T_Levin(x) = O(2^{Kt(x)})

Il lato ottimistico: l'algoritmo è *asintoticamente ottimale* — nessun altro algoritmo può trovare la soluzione in tempo significativamente inferiore su tutti i problemi. Il lato pessimistico: il fattore 2^{Kt(x)} è esponenziale in Kt(x), il che rende l'algoritmo praticamente inutilizzabile per problemi con Kt elevata. L'ottimalità asintotica di Levin è la risposta formale alla domanda "qual è il miglior algoritmo di ricerca possibile?" — ed è una risposta che scoraggia.

Per la cognizione, questo significa che i problemi con alta Kt — quelli per cui nessuna rappresentazione concisa e veloce esiste — non hanno soluzioni praticamente raggiungibili, non per difetto dell'agente ma per struttura del problema. Vi sono domini in cui la comprensione profonda è strutturalmente inaccessibile in tempo ragionevole, non perché i dati manchino ma perché il programma ottimale richiede un tempo di esecuzione che supera qualsiasi orizzonte temporale cognitivamente rilevante.

IV.4 Il mondo genuinamente casuale: compressione come illusione

Il secondo limite è di natura epistemica piuttosto che computazionale. Vi sono processi nel mondo che sono genuinamente stocastici — non solo apparentemente irregolari ma incomprimibili in linea di principio perché la loro fonte è rumore quantistico o caos deterministico ai limiti della sensibilità misurabile. Per tali processi, la ricerca di una struttura compressa non è un programma di ricerca: è la produzione di un'illusione.

L'illusione ha un nome nella psicologia cognitiva: *apofenia*, la tendenza a percepire pattern significativi in dati casuali. L'apofenia non è una patologia marginale — è una conseguenza strutturale dell'ottimizzazione per KC bassa in un sistema che non dispone di un test affidabile per la genuina casualità dell'input. Un sistema cognitivo addestrato a trovare struttura troverà struttura anche dove non ne esiste, producendo rappresentazioni con KC bassa che non corrispondono ad alcuna struttura reale del dominio.

Il problema formale è il seguente. Dato un campione finito x di una sorgente, è impossibile determinare con certezza se la sorgente è casuale o strutturata ma con struttura che si manifesta solo su campioni più lunghi. Per il teorema di incompletezza in forma computazionale (la non-calcolabilità di KC), nessun test statistico finito può escludere l'esistenza di una struttura compressa più lunga del campione disponibile. La distinzione tra "genuinamente casuale" e "strutturato oltre l'orizzonte di osservazione" è indecidibile.

Questo pone un limite fondamentale alla razionalità della compressione come strategia cognitiva: non si può sapere quando smettere di cercare struttura. Il principio del rasoio di Occam — preferire le spiegazioni più semplici — è razionale in attesa di evidenza contraria, ma non è giustificato da alcun teorema che garantisca che la struttura più semplice compatibile con i dati sia la struttura reale. La compressione cognitiva è una scommessa indimostrabilmente vincente.

IV.5 Sistemi con KC strutturalmente alta: i domini refrattari

Tra i limiti computazionali e l'illusione epistemica si collocano i *domini refrattari*: aree di fenomeni che non sono incomprimibili per ragioni di casualità, ma la cui struttura profonda ha KC intrinsecamente alta — non perché nessuna regolarità esista, ma perché la regolarità richiede descrizioni lunghe che non si abbreviano.

L'esempio più chiaro è quello dei sistemi complessi con interazioni a molti corpi. La meccanica statistica fuori dall'equilibrio, la turbolenza in fluidodinamica, l'ecologia delle comunità biologiche: questi sono domini in cui esistono leggi — l'equazione di Boltzmann, le equazioni di Navier-Stokes, le dinamiche di competizione di Lotka-Volterra — ma queste leggi hanno un KC del comportamento effettivo del sistema che cresce rapidamente con il numero di variabili e con l'orizzonte temporale. La descrizione ottimale di una traiettoria turbolenta su un orizzonte lungo è vicina alla KC massima: il sistema è, in questa accezione, quasi-casuale, ma per ragioni dinamiche e non stocastiche.

In questi domini, la strategia del pensiero come compressione produce leggi valide ma con capacità predittiva decrescente: le equazioni di Navier-Stokes descrivono correttamente la turbolenza ma non la prevedono in dettaglio oltre l'orizzonte di Lyapunov. Il guadagno di KC ottenuto dalla comprensione teorica è reale, ma la profondità logica delle previsioni è intrinsecamente limitata dall'amplificazione del rumore iniziale.

IV.6 Il limite fenomenologico: la prima persona e l'hard problem

Il terzo limite è il più controverso, e richiede la massima cautela nell'affermazione di ciò che si sa davvero e di ciò che si congettura. Si tratta dell'esperienza soggettiva nel senso tecnico che David Chalmers ha reso celebre con la locuzione *hard problem of consciousness*: la domanda non perché esistano stati funzionali correlati a comportamenti specifici, ma perché questi stati abbiano una qualità fenomenica — perché vi sia "qualcosa che è come" vedere il rosso, sentire dolore, udire una melodia.

Il punto che si vuole sollevare qui non è la questione metafisica nel suo complesso — sarebbe presuntuoso — ma una domanda più specifica: la struttura dell'esperienza soggettiva è, in linea di principio, comprimibile? Esiste una descrizione ottimale dell'esperienza fenomenica che catturi la sua struttura in modo tale da consentire ricostruzione, previsione, generalizzazione?

La risposta negativa — che l'esperienza soggettiva sia strutturalmente resistente alla compressione — non dipende dalla sua semplicità o complessità, ma da un'asimmetria di prospettiva. La compressione, come si è definita in tutto questo lavoro, opera su oggetti che possono essere specificati indipendentemente dalla prospettiva dell'osservatore: una stringa, un programma, un insieme di dati. L'esperienza fenomenica, per definizione, è *prospettiva-dipendente*: non esiste una descrizione di terza persona dell'esperienza di vedere il rosso che catturi ciò che essa è per chi la vive,

perché ciò che la rende quella esperienza specifica è precisamente la sua collocazione nella prospettiva di prima persona.

In termini formali, se si accetta che ogni descrizione D(e) dell'esperienza e sia una stringa su un alfabeto finito, allora D(e) è in principio comprimibile. Il problema non è la lunghezza della descrizione ma la sua *adeguatezza*: nessuna stringa D(e), per quanto lunga o concisa, cattura ciò che rende e un'esperienza — la sua qualità fenomenica — perché quella qualità non è un oggetto descritto ma un atto di descrizione, o più precisamente, l'atto del vivere che si descrive.

Questo non è un argomento definitivo — il dibattito sull'hard problem è aperto e la posizione qui adottata è quella di chi riconosce il problema senza pretendere di risolverlo. Ma è un argomento che delinea il confine del programma: il pensiero come compressione è un modello ricco e formalmente fondato per la cognizione che opera su oggetti di terza persona — teorie, percezioni, categorie, strutture causali. Diventa incompleto quando si incontra la prima persona, non perché essa sia inesplicabile in assoluto, ma perché i suoi oggetti non sono riducibili alla tipologia che la compressione presuppone.

IV.7 Una mappa dei limiti

I tre limiti si possono ora disporre in modo che la loro struttura reciproca risulti intellegibile.

Il limite computazionale (indecidibilità di KC, esponenzialità dell'algoritmo di Levin) è un limite *interno* al programma: opera all'interno dello stesso quadro formale e ne individua i confini di praticabilità. Non invalida il programma ma ne delimita l'applicabilità: la compressione ottimale è irraggiungibile, ma le sue approssimazioni sono computabili, e la distanza dall'ottimo è una misura sensata di quanto lontano un agente sia dalla comprensione piena.

Il limite epistemico (apofenia, indecidibilità della casualità) è un limite *trasversale*: riguarda il rapporto tra la strategia cognitiva e il mondo a cui essa è applicata. Il pensiero come compressione è razionale sotto l'assunzione che il mondo abbia struttura — ma non può dimostrare dall'interno quella struttura. È una scommessa sulla comprimibilità del reale che nessun argomento puramente formale può giustificare completamente: dipende, in ultima istanza, da assunti metafisici che la teoria non può fondare.

Il limite fenomenologico (hard problem) è un limite *esterno*: riguarda un dominio — l'esperienza soggettiva di prima persona — che non è correttamente modellato dagli oggetti su cui la compressione opera. Non è che la compressione fallisca quando applicata all'esperienza: è che applicarla presuppone una traduzione che potrebbe non essere disponibile.

Questi tre limiti non demoliscono il programma: lo circoscrivono. Una teoria del pensiero come compressione — fondata sul triangolo KC–D–Kt, articolata attraverso la fenomenologia dell'insight come transizione di fase, differenziata rispetto ai tipi di

compressione realizzati da sistemi umani e artificiali — è una delle ipotesi più rigorose e fruttuose disponibili per la cognizione nei domini che abitiamo di solito: la scienza, la matematica, la pianificazione, la comprensione linguistica, la percezione. Riconoscere i suoi limiti non è umiltà retorica: è la condizione perché essa rimanga, nei domini in cui si applica, una tesi e non un dogma.

Nota conclusiva: Profondità e dignità del comprendere

C'è una ragione per cui la distinzione tra KC e D — tra economia sintattica e storia computazionale sedimentata — merita di essere presa sul serio al di là della tecnicità. Essa cattura qualcosa che l'intuizione comune sul comprendere ha sempre saputo senza saperlo dire con precisione: che capire non è semplicemente sapere, non è disporre di un resoconto breve, non è produrre un output corretto. È avere incorporata nella propria struttura cognitiva la storia del processo che ha generato quella comprensione — i tentativi falliti, le revisioni, le anomalie assorbite, le distinzioni faticosamente conquistate.

Un sistema che produce output corretti senza portare quella storia è utile, talvolta straordinariamente utile. Ma non è, nel senso preciso che Bennett ha reso misurabile, *profondo*. La profondità logica non è una metafora della dignità intellettuale: è una misura del costo computazionale irriducibile che trasforma l'informazione in comprensione, la risposta in sapere. Che essa sia difficile da raggiungere, costosa da mantenere, impossibile da calcolare con precisione — tutto questo non la rende meno reale. La rende, semmai, più degna di attenzione.

Capitolo V — Nodi teorici: sofisticazione, complessità effettiva e i limiti della profondità

V.1 Un quarto vertice: la sofisticazione di Koppel

Il triangolo KC–D–Kt che ha sorretto l'impalcatura dei capitoli precedenti lascia fuori una quarta misura che ne costituisce, per certi versi, il completamento naturale: la *sofisticazione*, introdotta da Moshe Koppel nel 1988 e sviluppata in collaborazione con Henri Atlan nel contesto della complessità biologica. La sofisticazione non si aggiunge al triangolo come quarto vertice indipendente — è legata alle altre tre da relazioni precise — ma cattura un aspetto che nessuna delle tre misure precedenti coglie direttamente: la distinzione tra struttura *accidentale* e struttura *necessaria* in un oggetto.

L'idea di fondo può essere formulata così. Data una stringa x, la sua descrizione ottimale di lunghezza K(x) può essere decomposta in due parti: un *modello* p — la parte che cattura la struttura non accidentale di x — e dei *dati* d — la parte che cattura le specificità irriducibili, l'informazione contingente che il modello non può generare

autonomamente. La sofisticazione di x è la lunghezza della parte di modello nella decomposizione ottimale.

Formalmente, si dice che una coppia (p, d) è una *buona teoria* per x se U(p, d) = x — il programma p applicato ai dati d produce x — e se K(x | p) è grande, ovvero se x rimane incomprimibile anche dopo che p è noto. Quest'ultima condizione è cruciale: se K(x | p) fosse piccola, p da solo spiegherebbe x quasi completamente, e d sarebbe ridondante. Viceversa, se K(x | p) ≈ K(x), allora p non spiega niente, e tutta la struttura di x è nella sua specificità accidentale. La buona teoria è quella in cui p cattura esattamente la struttura regolare, e d contiene solo il residuo essenzialmente casuale.

La *sofisticazione* di x al livello di significatività c è:

$$\text{soph}_c(x) = \min\{\, |p| : \exists \text{ d tale che } U(p, d) = x \text{ e } K(x \mid p) \geq K(x) - c \,\}$$

Il minimo è preso su tutti i programmi p tali che esista un d con cui p genera x, e tali che x sia ancora "quasi incomprimibile" dato p — il che significa che p cattura la struttura regolare ma non l'informazione contingente. Il parametro c controlla la tolleranza: per c piccolo si richiedono teorie che non spieghino quasi nulla da sole; per c grande si ammettono teorie parzialmente esplicative.

V.2 Le tre classi rivisitate attraverso la sofisticazione

La sofisticazione riorganizza la classificazione degli oggetti in modo diverso e complementare rispetto a KC e D.

Una stringa casuale x con KC(x) ≈ n ha sofisticazione bassa. La ragione è che per essa il modello ottimale è il programma vuoto (o quasi): p = ε, d = x. La condizione K(x | p) ≥ K(x) - c è soddisfatta con p triviale, perché x è casuale e non ha struttura che un modello possa catturare. Dunque soph(x) ≈ 0: le stringhe casuali, pur avendo KC massima, hanno sofisticazione minima. Non hanno struttura necessaria — solo contingenza.

Una stringa banale come 0^n ha anch'essa sofisticazione bassa, ma per ragioni diverse. Il modello p = "stampa 0 per n volte" ha lunghezza O(log n) e K(0^n | p) ≈ 0: dato il modello, la stringa è completamente determinata, senza residuo casuale. Poiché la condizione di buona teoria richiede K(x | p) ≥ K(x) - c, e K(0^n) ≈ log n, l'unica buona teoria è quella con |p| ≈ 0 — quasi nessun modello, perché la stringa banale non ha struttura non-accidentale. Dunque soph(0^n) ≈ 0.

Le stringhe con alta sofisticazione sono quelle che richiedono un modello lungo per catturare la loro struttura non-accidentale, e in cui il residuo dopo il modello è genuinamente casuale. Un genoma di un organismo complesso è l'esempio paradigmatico: il modello p che codifica l'organizzazione funzionale del genoma — la struttura dei geni, dei promotori, dei segnali di splicing, delle reti di regolazione — è lungo, e il residuo dopo quel modello (le varianti neutrali, i siti polimorfici) è

approssimativamente casuale. La sofisticazione del genoma è alta, e corrisponde intuitivamente alla sua complessità *organizzata*: non semplicemente la quantità di informazione che contiene, ma la quantità di struttura funzionale irriducibile.

V.3 La relazione tra sofisticazione e profondità

Sofisticazione e profondità logica sono correlate ma non equivalenti, e la loro differenza è concettualmente importante.

La profondità D misura il tempo computazionale richiesto dal programma ottimale per generare x. La sofisticazione soph misura la lunghezza della parte strutturalmente necessaria di quel programma. Le due misure possono divergere in casi significativi.

Consideriamo un oggetto con alta D e bassa soph: un programma breve che compie un calcolo lungo ma producendo una stringa essenzialmente casuale (come i decimali di una funzione pseudocasuale crittograficamente sicura). Il programma è breve (KC bassa), il calcolo è lungo (D alta), ma il risultato non ha struttura catturabile da un modello (soph bassa). Tale oggetto è computazionalmente profondo ma non sofisticato — è il risultato di un processo lungo che non produce organizzazione.

Consideriamo viceversa un oggetto con alta soph e D moderata: un testo scientifico complesso, come un trattato di meccanica quantistica. Il modello che cattura la sua struttura non-accidentale — le equazioni, i formalismi, le dimostrazioni — è lungo (soph alta), ma il processo di generazione del testo stesso, una volta che il modello è noto, può essere relativamente rapido (D moderata). Tale oggetto è sofisticato ma non massimamente profondo nel senso di Bennett.

La relazione formale tra le due misure può essere derivata osservando che $\text{soph}_c(x) \leq K(x)$ per definizione — il modello non può essere più lungo della descrizione ottimale di x — e che $D_s(x)$ non è direttamente bounded da soph. Si può tuttavia dimostrare che oggetti con soph alta tendono ad avere D moderata o alta: se il modello p è lungo, il programma (p, d) che genera x deve "usare" p in modo non triviale, il che tipicamente richiede un tempo di esecuzione superiore a quello di un programma che non incorpora struttura organizzativa. Ma questo è un'osservazione qualitativa, non un teorema stringente.

V.4 La complessità effettiva di Gell-Mann e Lloyd

Indipendentemente dalla sofisticazione di Koppel, Murray Gell-Mann e Seth Lloyd hanno proposto negli anni Novanta la nozione di *complessità effettiva* come misura della quantità di informazione contenuta nelle regularità di un sistema — escludendo l'informazione contingente e irriducibile. La complessità effettiva EC(x) di un oggetto x è, informalmente, la lunghezza della descrizione compressa del suo schema regolare — non del singolo oggetto ma della classe di oggetti di cui x è un'istanza.

La definizione precisa richiede di specificare una distribuzione di probabilità sull'insieme degli oggetti considerati, e la complessità effettiva è allora la lunghezza della descrizione più concisa di quella distribuzione che sia compatibile con i dati

osservati. Formalmente, se $\mathscr{P}$ è una classe di distribuzioni di probabilità su $\mathscr{V}^*$, la complessità effettiva di x rispetto a $\mathscr{P}$ è:

$$EC(x, \mathscr{P}) = \min\{ K(P) : P \in \mathscr{P}, -\log_2 P(x) \leq K(x) + c \}$$

dove la condizione $-\log_2 P(x) \leq K(x) + c$ richiede che il modello P assegni a x una probabilità non troppo inferiore alla probabilità algoritmica ottimale. Il minimo è preso sull'insieme dei modelli compatibili.

La complessità effettiva è zero per oggetti casuali (nessun modello li descrive meglio della distribuzione uniforme, che ha K bassa ma non cattura nulla di specifico di x) e zero per oggetti banali (il modello è triviale, ma anche l'oggetto non ha struttura interna interessante). È massima per oggetti che richiedono modelli lunghi per descrivere le loro regolarità — esattamente come la sofisticazione.

Il rapporto tra EC e soph non è identità: le definizioni differiscono in modo tecnico nella scelta dell'insieme di riferimento e nel trattamento del parametro di tolleranza. Ma le due misure catturano la stessa intuizione fondamentale: la complessità interessante — quella che corrisponde all'organizzazione, alla struttura funzionale, alla profondità semantica — non è né il caos né la banalità, ma la zona intermedia in cui esistono regolarità lunghe da descrivere.

V.5 Il contributo di Atlan: complessità e autonomia biologica

Henri Atlan ha sviluppato, a partire dal suo lavoro degli anni Settanta sulla teoria dell'informazione nei sistemi biologici, una prospettiva sulla complessità che converge con la sofisticazione di Koppel da una direzione diversa. Atlan osservò che l'entropia di Shannon, applicata ai sistemi biologici, cattura la loro variabilità ma non la loro organizzazione funzionale: un sistema biologico degradato (un organismo morente) può avere entropia alta quanto un sistema organizzato, perché l'entropia misura la variabilità dell'output indipendentemente dalla sua struttura.

La nozione che Atlan sviluppò — in collaborazione con Koppel e con il progetto della *complessità organizzata* — è quella di un sistema che genera la propria organizzazione internamente, senza che essa sia interamente specificata dall'esterno. Un sistema è *autonomo* nel senso di Atlan se la parte del suo comportamento spiegata da un modello esterno (l'equivalente del programma p nella sofisticazione) è piccola rispetto alla parte che emerge dall'interazione interna dei componenti. L'autonomia è, in questo senso, complementare alla sofisticazione: un sistema altamente autonomo ha una sofisticazione che non può essere catturata da modelli di terza persona.

Il collegamento con il limite fenomenologico discusso nel Capitolo IV è diretto: la difficoltà di comprimere l'esperienza soggettiva può essere reinterpretata, nel linguaggio di Atlan, come un caso estremo di autonomia — la soggettività è il sistema che non può essere descritto dall'esterno senza perdita irriducibile, perché la sua organizzazione è generata dall'interno in modo che nessun modello esterno cattura completamente.

V.6 Il quadrilatero: soph, KC, D, Kt e la topologia della comprensione

Con l'aggiunta della sofisticazione, la mappa dello spazio cognitivo si arricchisce di una dimensione. Il quadrilatero KC–D–Kt–soph permette di distinguere quattro tipi di oggetti cognitivamente rilevanti che il triangolo lasciava confusi.

Gli oggetti *profondi e sofisticati* hanno KC bassa, D alta e soph alta: richiedono un modello lungo, quel modello deve essere eseguito a lungo per produrre l'oggetto, e il risultato porta struttura non-accidentale genuina. La comprensione di un sistema biologico complesso — diciamo, il ciclo di Krebs nel contesto del metabolismo cellulare — appartiene a questa classe: il modello è lungo (la biochimica del ciclo è elaborata), il processo di comprensione è costoso (si impara attraverso un lungo percorso di studio), e la struttura catturata è funzionalmente organizzata, non accidentale.

Gli oggetti *profondi ma non sofisticati* hanno KC bassa, D alta e soph bassa: il programma che li genera è breve e lento, ma produce un output privo di struttura organizzativa catturabile. Certi processi crittografici appartengono a questa classe: il programma è conciso e il calcolo lungo, ma l'output è progettato per essere statisticamente indistinguibile dal rumore.

Gli oggetti *sofisticati ma non profondi* hanno soph alta e D moderata: la struttura non-accidentale è ricca, ma il processo di generazione è relativamente rapido. Un testo in lingua naturale su un argomento tecnico appartiene spesso a questa classe: la struttura sintattica, semantica e pragmatica è organizzata (soph alta), ma produrre il testo — una volta che il modello linguistico è acquisito — non richiede un calcolo particolarmente lungo (D moderata).

Gli oggetti *banali* hanno tutte e quattro le misure basse: descrizione breve, generazione rapida, struttura non-accidentale assente. I dati di una tavola di verità di una funzione booleana semplice appartengono a questa classe.

V.7 Complessità resource-bounded e la cognizione come compressione vincolata

Un nodo teorico che merita attenzione separata è quello della *complessità resource-bounded*: la famiglia di misure che generalizzano KC imponendo vincoli espliciti sulle risorse computazionali disponibili — tempo, spazio, energia — e studiano come cambia la struttura della compressione ottimale sotto questi vincoli.

La complessità resource-bounded più studiata è la complessità di Kolmogorov limitata nel tempo, spesso indicata $K^t(x)$ (distinta dalla Kt di Levin per la presenza del superscript):

$$K^t(x) = \min\{ |p| : U(p) = x \text{ e } t(p) \leq t \}$$

dove t è un bound sul tempo di esecuzione. K^t(x) misura la lunghezza del programma più breve che genera x entro t passi: all'aumentare di t, K^t(x) decresce monotonicamente fino a K(x). La differenza K^t(x) - K(x) misura il *costo della velocità*: quanto ci si paga in lunghezza di descrizione per vincolare il tempo di calcolo.

Per la cognizione, questa misura cattura una verità che l'esperienza rende immediata: non si ha mai accesso alla compressione ottimale, bensì alla compressione ottimale compatibile con i vincoli cognitivi disponibili — tempo di elaborazione, capacità di memoria, energia metabolica del sistema nervoso. La cognizione umana opera sotto un bound t che, per la maggior parte dei compiti quotidiani, è dell'ordine di qualche centinaio di millisecondi (il tempo di una decisione percettiva) o di qualche decina di secondi (il tempo di una decisione deliberativa). Per compiti che richiedono riflessione prolungata, t può estendersi a ore, giorni, anni.

Il costo della velocità K^t - K misura quanto "capire in fretta" peggiori la qualità della comprensione. Questo ha un analogo diretto nell'errore cognitivo di sistema-1 vs sistema-2 nella terminologia di Kahneman: il sistema-1 opera con un bound t molto basso e paga un costo elevato in lunghezza descrittiva — usa euristiche brevi ma non ottimali; il sistema-2 opera con t più alto e si avvicina alla compressione ottimale, al costo di un impegno cognitivo maggiore.

La differenza $K^{t_1}(x) - K^{t_2}(x)$ per $t_1 < t_2$ misura quanto migliora la comprensione aumentando il tempo di elaborazione. Per certi problemi, questa differenza è piccola — il sistema-1 fa quasi altrettanto bene del sistema-2, e il tempo aggiuntivo non produce guadagno significativo. Per altri problemi, la differenza è grande — problemi che richiedono genuinamente elaborazione profonda, dove la fretta produce descrizioni molto subottimali. L'identificazione di quali compiti appartengono a quale categoria è, in questo senso, una questione empirica che la complessità resource-bounded formalizza ma non risolve da sola.

V.8 Il problema dell'invarianza e la dipendenza dalla macchina universale

Un nodo teorico che ogni trattazione onesta di KC è tenuta ad affrontare è quello dell'*invarianza*: tutte le misure discusse — KC, D, Kt, soph — dipendono dalla scelta della macchina di Turing universale U, e cambiano al variare di U. Il teorema di invarianza assicura che questa dipendenza è limitata a una costante additiva:

$$K_U(x) - K_V(x) = O(1)$$

per qualsiasi coppia di macchine universali U e V, dove la costante dipende dalla coppia ma non da x. Analogamente per D e Kt.

L'invarianza è rassicurante per le applicazioni asintotiche — classificare oggetti come "profondi" o "superficiali" non dipende dalla scelta di U per stringhe sufficientemente lunghe — ma problematica per le applicazioni cognitive, dove si confrontano oggetti

di lunghezza finita e la costante di invarianza può essere significativa rispetto alle differenze che si vogliono misurare.

Per la cognizione, questo si traduce in un fatto che la teoria non può aggirare: le misure KC, D, Kt, soph sono definite rispetto a un linguaggio di descrizione — equivalentemente, rispetto a un sistema di rappresentazione. Agenti cognitivi diversi, che usano sistemi di rappresentazione diversi, possono assegnare valori diversi a KC(x) o soph(x) per lo stesso oggetto x, con differenze che non sono riducibili a costanti additive se i sistemi di rappresentazione differiscono in modo strutturale (per esempio, se uno è di ordine zero e l'altro è di ordine superiore).

Questo non invalida l'uso delle misure come strumenti teorici, ma suggerisce che quando si dice "un esperto ha una rappresentazione con KC più bassa di quella di un novizio" si sta implicitamente assumendo un sistema di rappresentazione condiviso rispetto al quale entrambe le descrizioni vengono misurate. L'apprendimento di un dominio include, in parte, l'acquisizione del sistema di rappresentazione adeguato — il linguaggio nel quale la compressione è possibile. Chi non ha ancora acquisito il sistema di rappresentazione della meccanica quantistica non può avere una bassa KC della meccanica quantistica, non perché sia meno intelligente, ma perché la sua U non è equipaggiata con le subroutine necessarie per trattare quella classe di oggetti.

Questo punto connette la teoria formale con la pedagogia: insegnare un dominio è, in parte significativa, installare nell'apprendente il sistema di rappresentazione rispetto al quale il dominio diventa comprimibile. Prima che quel sistema sia operativo, la comprensione del dominio non è bassa KC rispetto a una U inadeguata: è assenza di KC sensata.

Capitolo VI — La compressione a due agenti: gap di rappresentazione e costo dell'incomprensione

---

VI.1 Il problema della comunicazione tra macchine diverse

L'intera impalcatura dei capitoli precedenti riposa su un'assunzione mai esplicitata, che merita ora di essere portata alla luce e interrogata: che esista una macchina di Turing universale U rispetto alla quale tutte le misure — KC, D, Kt, soph — sono definite, e che questa U sia in qualche modo condivisa dagli agenti cognitivi di cui si discute. Il teorema di invarianza (Appendice A) garantisce che la scelta di U non altera la classificazione degli oggetti a meno di costanti additive, per stringhe sufficientemente lunghe. Ma questo garantisce solo che le misure siano ben definite *in astratto*: non che due agenti reali, con sistemi di rappresentazione strutturalmente diversi, misurino KC, D o soph degli stessi oggetti in modo compatibile.

Il problema che si vuole affrontare in questo capitolo è il seguente: cosa accade, nel quadro formale della monografia, quando due agenti cognitivi — chiamiamoli α e β —

adoperano sistemi di rappresentazione che non sono semplicemente due macchine universali U_α e U_β legate da una costante additiva, ma che differiscono in modo *strutturale*? Il termine "strutturale" va precisato: si dirà che due macchine U_α e U_β differiscono strutturalmente se la simulazione di U_α da parte di U_β non è realizzabile con un programma di simulazione s di lunghezza costante, ma richiede un programma la cui lunghezza cresca con la complessità degli oggetti da trattare. Questa situazione — apparentemente esotica dal punto di vista della teoria della computazione, dove tutte le macchine universali sono simulate l'una dall'altra a costo costante — emerge naturalmente nella cognizione reale, dove i "sistemi di rappresentazione" sono costrutti storicamente determinati, parziali e non universali nel senso tecnico.

VI.2 Macchine di rappresentazione parziali e il gap di simulazione

Un agente cognitivo reale non è una macchina di Turing universale: è un sistema con un repertorio finito e specifico di primitive computazionali. Un matematico del XVII secolo aveva a disposizione geometria euclidea, aritmetica, e la logica sillogistica aristotelica — un insieme di primitive che chiamiamo U_{XVII}. Un matematico contemporaneo dispone di quelle stesse primitive più il calcolo infinitesimale, la teoria degli insiemi, la topologia, la teoria delle categorie — un insieme di primitive U_{XXI}. La differenza non è che U_{XXI} sia "più veloce" o "più grande" di U_{XVII}: è che U_{XXI} contiene primitive che in U_{XVII} richiederebbero programmi di lunghezza arbitrariamente crescente per essere simulate.

Si formalizzi questa idea. Sia P_α l'insieme delle primitive computazionali dell'agente α — le operazioni elementari che α può eseguire in un singolo passo. Si definisce la macchina di rappresentazione di α come la macchina di Turing M_α costruita su P_α: ogni programma in M_α è una sequenza di operazioni in P_α, e il costo di ogni operazione è unitario. M_α non è in generale universale nel senso tecnico: potrebbe non essere in grado di simulare tutte le macchine di Turing. Ma per la classe di oggetti cognitivamente rilevanti — le stringhe che rappresentano conoscenza di un dominio — M_α può essere sufficiente.

Il *gap di simulazione* tra due agenti α e β è definito come:

$$\mathrm{Gap}(\alpha, \beta; x) = K_{M_\alpha}(x) - K_{M_\beta}(x)$$

dove $K_{M_\alpha}(x)$ è la complessità di Kolmogorov di x rispetto alla macchina M_α, e analogamente per β. Questo gap misura di quanti bit la descrizione ottimale di x da parte di α eccede quella da parte di β: è positivo se β ha una rappresentazione più concisa di x, negativo se α ce l'ha.

A differenza del caso delle macchine universali — dove il gap è bounded da una costante che dipende solo dalle macchine e non da x — per macchine di rappresentazione parziali il gap può crescere con la complessità di x. Se β dispone di una primitiva che α non ha, e quella primitiva è esattamente ciò che serve per

comprimere x, allora Gap(α, β; x) cresce arbitrariamente con la profondità della struttura di x.

VI.3 Il costo della comunicazione tra paradigmi

Si consideri ora la situazione comunicativa: α vuole trasmettere a β la propria comprensione di un oggetto x. La comprensione di α è il programma p_α — il programma ottimale per x rispetto a M_α. Per trasmettere questa comprensione, α deve codificare p_α in un linguaggio che β possa decodificare. Il costo di questa trasmissione è la lunghezza della descrizione di p_α rispetto a M_β: non la lunghezza del programma ottimale per x in M_β, ma la lunghezza della *descrizione del programma di a* nel linguaggio di β.

Questa quantità, che chiamiamo *costo di traduzione* da α a β per l'oggetto x, è:

$$CT(\alpha \to \beta; x) = K_{M_\beta}(p_\alpha)$$

dove p_α è il programma ottimale di α per x. Il costo di traduzione non è simmetrico: $CT(\alpha \to \beta; x) \neq CT(\beta \to \alpha; x)$ in generale, e la direzione della comunicazione ha costi strutturalmente diversi.

Si notino tre casi limite che chiariscono il significato della misura.

Se M_α e M_β condividono le stesse primitive, p_α è direttamente eseguibile su M_β e $CT(\alpha \to \beta; x) = |p_\alpha| + O(1)$: il costo di traduzione è essenzialmente la lunghezza del programma di α, con overhead costante. La comunicazione è piena e senza perdita.

Se M_β contiene le primitive di M_α più primitive aggiuntive, il programma p_α è ancora eseguibile su M_β, ma β può disporre di un programma p_β più corto per x. In questo caso $CT(\alpha \to \beta; x) = |p_\alpha| + O(1)$ come prima, ma $Gap(\alpha, \beta; x) > 0$: β ha una rappresentazione migliore, ma può ricevere e decodificare quella di α. La comunicazione è piena ma subottimale per β.

Se M_α contiene primitive che M_β non ha, allora p_α non è direttamente eseguibile su M_β. Per tradurlo, α deve sostituire ogni occorrenza delle proprie primitive non-β con subroutine espresse in M_β, e il costo di questa sostituzione può essere arbitrariamente alto. Se la primitiva non disponibile è quella che permette di comprimere x, ogni subroutine che la simula ha lunghezza crescente con la profondità di x, e $CT(\alpha \to \beta; x)$ cresce senza limite.

VI.4 L'incomprensione strutturale come fenomeno formale

Il terzo caso limite del paragrafo precedente è la formalizzazione dell'*incomprensione strutturale*: la situazione in cui α non riesce a trasmettere la propria comprensione a β non per mancanza di buona volontà o di chiarezza espositiva, ma perché il sistema di

rappresentazione di β non dispone delle primitive necessarie per ricevere il messaggio senza una perdita che cresce con la profondità dell'oggetto.

Questo fenomeno è radicalmente diverso dall'ignoranza ordinaria. L'ignoranza ordinaria è la situazione in cui β non ha ancora trovato il programma p_β ottimale per x — ma M_β lo ammette in linea di principio. L'incomprensione strutturale è la situazione in cui anche il programma ottimale di β per x è radicalmente peggiore di quello di α, per ragioni che non dipendono dallo sforzo cognitivo di β ma dalla struttura di M_β.

Si può dare una stima del *grado di incomprensione strutturale* di β rispetto a α per l'oggetto x:

$$IS(\alpha, \beta; x) = CT(\alpha \to \beta; x) - K_{M_\beta}(x)$$

Questa quantità misura l'overhead che β deve pagare per capire la comprensione di α rispetto al costo di trovare autonomamente la propria rappresentazione ottimale. Se IS è piccolo, la comprensione di α è un aiuto per β: il programma di α, tradotto, è quasi ottimale anche in M_β. Se IS è grande, la comprensione di α è quasi inutile per β: β farebbe meglio a costruirsi la propria rappresentazione da zero piuttosto che cercare di tradurre quella di α.

La storia della comunicazione scientifica tra paradigmi è piena di situazioni corrispondenti a IS elevata. I tentativi di Leibniz di spiegare il calcolo infinitesimale ai matematici che disponevano solo della geometria greca non producevano comprensione ma parafasi sempre più lunghe e sempre meno fedeli — esattamente la struttura di un $CT(\alpha \to \beta)$ che cresce con la profondità dell'oggetto. La meccanica quantistica rimane incomprensibile a chi non dispone delle primitive della meccanica hamiltoniana e dell'algebra lineare — non perché i testi siano oscuri, ma perché CT(fisico → non-fisico; QM) è strutturalmente elevato.

VI.5 Profondità congiunta e la misura dell'incomprensione condivisa

Si introduce ora una misura che cattura non l'incomprensione di β rispetto ad α, ma l'incomprensione *reciproca* tra due agenti — la struttura che nessuno dei due riesce a vedere con i propri strumenti.

Si definisce la *profondità congiunta* di due agenti α e β rispetto a un oggetto x come:

$$D_{joint}(\alpha, \beta; x) = D_{s}^{M_U}(x) - \max(D_{s}^{M_\alpha}(x), D_{s}^{M_\beta}(x))$$

dove M_U è la macchina universale rispetto alla quale la profondità "vera" di x è misurata, e $D_s^{M_\alpha}$ è la profondità di x rispetto alla macchina di α. La quantità

D_joint misura la profondità che x *avrebbe* se le primitive di entrambi i sistemi di rappresentazione fossero disponibili, meno la profondità che il più attrezzato dei due riesce a vedere.

Quando D_joint è grande, esiste struttura profonda in x che nessuno dei due agenti riesce a rappresentare adeguatamente con i propri strumenti. Questo è il caso delle collaborazioni scientifiche più fruttuose: fisici e matematici che lavorano su un problema portano sistemi di rappresentazione M_α e M_β con primitive complementari, e la struttura che emergono dalla collaborazione ha profondità superiore a quella accessibile a ciascuno autonomamente.

La misura D_joint suggerisce anche quando la collaborazione è strutturalmente sterile: se M_α e M_β hanno le stesse primitive — ovvero se uno simula l'altro a costo costante — allora D_joint è piccola e la collaborazione non produce guadagni di profondità irraggiungibili individualmente. La vera collaborazione intellettuale richiede differenza strutturale, non semplicemente più risorse dello stesso tipo.

VI.6 Il dialogo socratico come protocollo di riduzione del gap

L'analisi del gap di simulazione suggerisce una reinterpretazione del dialogo filosofico — in particolare nella sua forma socratica — come protocollo di comunicazione progettato per ridurre $CT(\alpha \to \beta)$ attraverso costruzione iterativa di primitive condivise.

Il metodo elenctico di Socrate non trasmette proposizioni: costruisce, attraverso una sequenza di domande e risposte, un sistema di primitive cognitive condivise — distinzioni concettuali, controesempi, generalizzazioni — che trasformano M_β in una macchina capace di ricevere la comprensione che Socrate intende comunicare. La lunga sequenza di domande preparatorie non è retorica: è il costo computazionale di costruire il sottoinsieme di M_α necessario per rendere il messaggio finale comprensibile.

In termini formali, il dialogo socratico è un protocollo che, attraverso n scambi comunicativi, riduce $CT(\alpha \to \beta; x)$ da un valore iniziale elevato a un valore finale vicino a $K_{M_\beta'}(x)$, dove M_β' è il sistema di rappresentazione arricchito di β dopo il dialogo. Il costo totale del dialogo — la lunghezza della trascrizione — è la somma dei costi dei singoli scambi, e il suo valore è giustificato quando la riduzione di CT è superiore a questo costo.

Quando invece il dialogo fallisce — quando il risultato è "non capire" nonostante la lunga conversazione — la spiegazione formale è che il dialogo non ha costruito primitive sufficienti: M_β' non è abbastanza diverso da M_β per rendere il messaggio finale ε-accessibile. Il fallimento del dialogo è un fenomeno topologico nel senso del Capitolo II: la barriera tra M_β e M_β' non è stata superata.

VI.7 Implicazioni per la comunicazione tra sistemi artificiali e umani

L'analisi del gap di rappresentazione ha implicazioni dirette per il problema della comunicabilità tra modelli di linguaggio e utenti umani — un problema che la letteratura corrente tratta quasi esclusivamente in termini di allineamento valoriale, trascurando la dimensione strutturale.

Un modello di linguaggio di grandi dimensioni ha un sistema di rappresentazione M_{LLM} determinato dalla sua architettura e dal suo addestramento. Come si è discusso nel Capitolo III, M_{LLM} è ottimizzato per bassa Kt e bassa D. Le primitive computazionali di M_{LLM} — i pattern di attivazione degli strati di attention — non coincidono con le primitive del sistema di rappresentazione umano, e la differenza non è riducibile a una costante additiva per oggetti sufficientemente profondi.

Questo genera un gap di simulazione strutturale: CT(LLM → umano; x) può essere elevato per oggetti x con alta profondità logica, non perché il modello non abbia "capito" x nel proprio spazio di rappresentazione, ma perché la sua rappresentazione di x — ottimale in M_{LLM} — non è traducibile a basso costo nel sistema di rappresentazione umano. Il risultato fenomenologico è la sensazione che il modello "dica le cose giuste ma non capisca davvero" — una diagnosi intuitiva che il gap di simulazione rende formalmente precisa: il modello ha bassa K_{M_{LLM}}(x) ma alto CT(LLM → umano; x).

La direzione opposta è altrettanto interessante: CT(umano → LLM; x) è bassa per oggetti con struttura linguistica ricca, e alta per oggetti la cui struttura è procedurale o spaziale — per la stessa ragione strutturale. Le primitive del sistema di rappresentazione umano che operano su strutture spaziali, temporali o causali non hanno corrispondenti diretti nelle primitive di M_{LLM}, e la loro traduzione nel linguaggio del modello produce overhead crescente con la profondità procedurale dell'oggetto.

Il pensiero come compressione

Capitolo VII — Il dimenticare come operatore di profondità

VII.1 La fenomenologia dell'oblio fecondo

C'è una categoria di esperienza intellettuale che ogni studioso conosce e che le teorie della memoria trattano sistematicamente male: il momento in cui ci si accorge di aver dimenticato quasi tutto di un libro o di un corso studiato anni prima, e di capire tuttavia il dominio *meglio* di quando lo si conosceva nei dettagli. Non si ricordano i nomi degli autori secondari, le date, le citazioni precise, le dimostrazioni step-by-step

— eppure il campo appare più trasparente, più strutturato, più navigabile di allora. Il dimentico sa dove guardare, riconosce le strutture rilevanti, valuta rapidamente l'importanza di un risultato nuovo. Ha perso informazione e guadagnato comprensione.

Questa esperienza è trattata nella letteratura cognitiva con imbarazzo, o ignorata del tutto: le teorie della memoria sono ottimizzate per spiegare il ricordo, non l'oblio produttivo. Le teorie dell'apprendimento trattano il dimenticare come deterioramento, rumore, fallimento del consolidamento. La psicologia della competenza esperta descrive l'esperto come chi "ricorda di più" rispetto al novizio — il che è vero per certi tipi di memoria, ma sistematicamente falso per altri.

Il quadro KC–D–Kt–soph permette di dare a questo fenomeno una spiegazione formale precisa, che non è una metafora del "vedere la foresta e non gli alberi" ma un teorema sulla struttura degli operatori di proiezione nello spazio delle descrizioni.

VII.2 Decomposizione della rappresentazione in modello e residuo

Il punto di partenza è la decomposizione che la sofisticazione di Koppel introduce: per ogni stringa x, la coppia ottimale (p, d) separa il *modello* p — la parte strutturalmente necessaria, che cattura le regolarità non-accidentali — dai *dati* d — il residuo contingente, l'informazione irriducibile che il modello non può generare. La lunghezza |p| è la sofisticazione di x; la lunghezza |d| è la parte accidentale.

Si definisce ora l'operatore di proiezione al modello come:

$\Pi_model : x \mapsto p_x$

che mappa ogni stringa x nel proprio modello ottimale p_x. L'immagine di Π_model è lo spazio dei modelli — le descrizioni delle regolarità non-accidentali degli oggetti. L'operatore complementare è la proiezione al residuo:

$\Pi_res : x \mapsto d_x$

che mappa x nel residuo accidentale d_x tale che U(p_x, d_x) = x.

Il dimenticare selettivo — nel senso fenomenologicamente rilevante — è l'applicazione parziale di Π_model: l'agente conserva p_x e perde d_x, o più precisamente, la sua rappresentazione si sposta da x a p_x — da "la stringa completa" al "modello della stringa". Le quattro misure si trasformano nel modo seguente sotto questa operazione:

$K(p_x) \leq K(x)$ (il modello è più conciso della stringa completa)

$D(p_x) \geq D(x)$ (il modello ha profondità maggiore: da dimostrare)

$Kt(p_x) \lesssim Kt(x)$ (indeterminato: dipende dal rapporto tra D e K)

$soph(p_x) \geq soph(x)$ (il modello è quasi interamente struttura non-accidentale)

La terza disuguaglianza è indeterminata e richiede analisi caso per caso. Le altre tre sono il contenuto principale di questo capitolo, e la seconda — $D(p_x) \geq D(x)$ — è il risultato meno ovvio e più rilevante.

VII.3 Il teorema dell'aumento di profondità per proiezione al modello

Enunciato. Sia (p_x, d_x) la decomposizione ottimale di x nel senso della sofisticazione, con $K(x|p_x) \geq K(x) - c$ per una costante c piccola. Allora, sotto condizioni che verranno precisate, vale:

$$D_s(p_x) \geq D_s(x) - O(|d_x|)$$

In parole: la profondità logica del modello p_x è almeno quella della stringa completa x, meno un termine proporzionale alla lunghezza del residuo accidentale.

Dimostrazione informale. La profondità $D_s(x)$ è il tempo minimo richiesto per generare x da un programma quasi-ottimale. Questo programma deve in qualche forma generare sia p_x che d_x, poiché $U(p_x, d_x) = x$. Il processo di generazione di x può essere scomposto in due fasi: generare p_x, poi usare p_x e d_x per produrre x. Se la seconda fase richiede tempo $T_2 = O(|d_x|)$ — proporzionale alla lunghezza del residuo da leggere e applicare — e la prima fase richiede tempo T_1, allora il tempo totale è $T_1 + T_2$.

Si ha:

$$D_s(x) \leq D_s(p_x) + O(|d_x|)$$

per costruzione (il programma ottimale per x può sempre essere decomposto come "genera p_x, poi applica d_x"). Riorganizzando:

$$D_s(p_x) \geq D_s(x) - O(|d_x|)$$

Quando $|d_x|$ è piccolo — quando il residuo accidentale è breve rispetto alla profondità di x — si ha $D_s(p_x) \approx D_s(x)$: il modello conserva quasi tutta la profondità della stringa originale. Quando $|d_x|$ è grande — quando c'è molta informazione accidentale — il bound è meno informativo. ▮

La condizione chiave è dunque che il residuo accidentale d_x sia breve. Questo è esattamente il caso degli oggetti cognitivamente interessanti: le regularità profonde di un dominio sono descritte da un modello lungo (soph alta) con un residuo accidentale relativamente breve (i dettagli contingenti che il modello non predice). Per tali oggetti, la proiezione al modello conserva quasi tutta la profondità e aumenta la sofisticazione — è un'operazione di guadagno cognitivo netto.

VII.4 Il dimenticare come purificazione strutturale

Il teorema precedente chiarisce perché il dimenticare i dettagli accidentali — le date, i nomi, le citazioni, i numeri specifici — produce spesso un aumento della comprensione e non una sua diminuzione. L'operazione cognitiva corrispondente è la proiezione Π_model: l'agente perde d_x ma conserva p_x, e poiché D_s(p_x) ≈ D_s(x), la profondità della sua rappresentazione rimane sostanzialmente intatta. Ciò che si perde è la capacità di recuperare i dettagli accidentali; ciò che si conserva — e talvolta si aumenta, come si mostrerà — è la capacità di ragionare sulla struttura.

La sofisticazione di p_x è invece strettamente maggiore di quella di x. Per definizione, soph(p_x) = |p_x| (il modello è quasi interamente struttura non-accidentale — il suo residuo ottimale è vuoto o quasi). Poiché soph(x) = |p_x| per costruzione, si ha:

soph(p_x) / K(p_x) > soph(x) / K(x)

Il *rapporto di sofisticazione* — la frazione della descrizione che è struttura non-accidentale — è più alto nel modello che nella stringa completa. Il modello è strutturalmente più puro: quasi tutta la sua lunghezza è informazione organizzativa, non contingente.

Questo cattura formalmente l'intuizione dell'esperto che "vede la struttura": la sua rappresentazione del dominio ha un rapporto di sofisticazione più alto di quella del novizio, non necessariamente perché sia più lunga, ma perché ha subito successive operazioni di proiezione al modello che hanno rimosso l'informazione accidentale e lasciato quella strutturale.

VII.5 L'oblio come successione di proiezioni: la dinamica della comprensione

Si può ora costruire un modello dinamico della comprensione come successione di operazioni di Π_model. Un agente che studia un dominio attraversa fasi distinguibili:

Nella fase di acquisizione, l'agente accumula x — la stringa completa, con modello e residuo. KC della sua rappresentazione cresce, ma soph e D rimangono basse: ha molti dettagli ma poca struttura estratta.

Nella fase di consolidamento, che avviene in parte durante il sonno (la letteratura sulla consolidazione mnestica notturna è consistente con questo modello) e in parte attraverso l'uso ripetuto, l'agente esegue implicitamente Π_model: i dettagli accidentali

vengono persi, e la rappresentazione converge verso p_x. KC scende, soph sale, D sale o rimane stabile.

Nella fase di competenza, l'agente opera principalmente su p_x. Quando incontra nuovi dati d_x' — nuovi dettagli dello stesso dominio — la sua operazione cognitiva è U(p_x, d_x'): applica il modello ai nuovi dati, produce l'output rilevante, e poi dimentica d_x' se non è strutturalmente informativo. Il ciclo di acquisizione-consolidamento-dimenticanza è diventato automatico.

Questo schema ha una conseguenza che la teoria della memoria standard fatica a spiegare: i novizi ricordano i dettagli meglio degli esperti (per certi tipi di test di richiamo verbatim), ma gli esperti *usano* il dominio molto più efficacemente. La spiegazione nel quadro KC–D–soph è che i novizi hanno x con residuo alto; gli esperti hanno p_x con residuo basso. I test di richiamo verbatim misurano la disponibilità di d_x — e gli esperti lo hanno perso. Ma i test di ragionamento, generalizzazione e trasferimento misurano la disponibilità di p_x e la profondità D_s(p_x) — e gli esperti le hanno entrambe.

VII.6 Il paradosso dell'incomprimibilità del modello ottimale

Si incontra ora un nodo sottile che il teorema precedente lascia aperto. Si è detto che $D_s(p_x) \geq D_s(x) - O(|d_x|)$, il che suggerisce che la profondità del modello è almeno quella della stringa. Ma può D_s(p_x) essere *strettamente maggiore* di D_s(x)?

La risposta è affermativa in casi significativi, e la ragione è la seguente. La stringa x può essere generata da un programma che "scorciatoia" la profondità: usa il residuo d_x come oracolo per saltare passi computazionali lunghi. In altre parole, il residuo accidentale può fungere da *istruzione* che abbrevia il calcolo del modello — una sorta di suggerimento che riduce il tempo di esecuzione. Quando questo accade, il programma ottimale per x ha tempo di esecuzione inferiore a quello del programma ottimale per p_x da solo, perché nel primo caso d_x è disponibile come risorsa.

Si dica che d_x è un *acceleratore computazionale* di p_x se:

$$D_s(x) < D_s(p_x) - \Omega(f(|p_x|))$$

per qualche funzione f crescente. Quando d_x è un acceleratore, dimenticare d_x *aumenta strettamente* la profondità logica della rappresentazione residua: l'agente deve ora lavorare più a lungo per generare lo stesso output, perché ha perso il suggerimento che abbreviava il calcolo.

Questo fenomeno ha una lettura cognitiva precisa. Certe informazioni "esterne" — un manuale di riferimento, una formula memorizzata, un esempio paradigmatico — abbreviano il calcolo cognitivo del modello: il modello applicato con quell'informazione a portata di mano produce output più rapidamente di quanto il modello "nudo" farebbe. Perdere quella informazione non indebolisce il modello, ma

lo obbliga a un cammino più lungo — più genuinamente derivativo, meno dipendente dall'esempio.

Il caso estremo è quello delle dimostrazioni matematiche: una dimostrazione della proposizione P è un d che, applicato al modello logico del sistema formale p, produce P come output in tempo breve. Un matematico che non ricorda la dimostrazione di un teorema che usa frequentemente deve *ricostruirla* ogni volta — un processo con D alta. Un matematico che la ricorda la recupera in tempo O(|d|). Il secondo ha KC più alta (perché ha d in memoria), ma D più bassa per la generazione di P. Il primo ha KC più bassa ma D più alta. Chi è il più "profondo" dei due? La risposta dipende dalla domanda: per *usare* P rapidamente, il secondo; per *comprendere* P nella sua struttura derivativa, il primo.

VII.7 La dimensione ottimale dell'oblio: un principio di minimizzazione

Si può ora formulare un principio che determina quale quantità di residuo un agente cognitivo dovrebbe ottimalmente dimenticare, in funzione delle proprie finalità.

Si supponga che l'agente voglia massimizzare D_s(p_x) — la profondità della propria comprensione strutturale — sotto il vincolo che il costo computazionale di applicare il modello a nuovi dati non superi una soglia τ:

massimizza D_s(p_{x'})

soggetto a Kt(U(p_{x'}, d_{new})) ≤ τ

dove p_{x'} è il modello dopo l'oblio parziale e d_{new} sono nuovi dati che l'agente deve elaborare. Il vincolo su Kt assicura che il modello residuo rimanga usabile entro il budget computazionale dell'agente.

La soluzione di questo problema di ottimizzazione è dimenticare tutto il residuo d_x che è un acceleratore computazionale *non necessario* — cioè che abbrevia il calcolo ma che il modello nudo può comunque compiere entro τ — e conservare il residuo che è strutturalmente necessario o che abbrevia calcoli altrimenti fuori dal budget τ. La dimensione ottimale dell'oblio è quella che massimizza la profondità strutturale compatibilmente con l'usabilità del modello.

Questo principio — che si potrebbe chiamare *principio del dimenticare ottimale* — ha implicazioni pedagogiche concrete. Un curriculum ottimale non è quello che massimizza la ritenzione dei dettagli né quello che si limita ai principi generali: è quello che costruisce il modello p_x con la profondità massima compatibile con la soglia di usabilità τ degli studenti, e che lascia dimenticare il residuo non strutturalmente necessario. I dettagli che andrebbero conservati sono quelli che sono acceleratori computazionali irrinunciabili — quelli senza i quali il modello non è usabile entro τ. I

dettagli che andrebbero lasciati dimenticare sono quelli che, pur presenti nel corpus originale, non contribuiscono né alla profondità del modello né alla sua usabilità.

Il pensiero come compressione

Capitolo VIII — Proof complexity come lower bound cognitivo

VIII.1 La proof complexity come teoria dei lower bound

La *proof complexity* è un settore della logica matematica e della teoria della complessità computazionale che studia non quanto sia difficile *verificare* una dimostrazione — operazione sempre fattibile in tempo polinomiale per sistemi formali standard — ma quanto debba essere *lunga* una dimostrazione di una certa proposizione in un dato sistema formale. La domanda fondamentale del campo è: esiste una tautologia proporzionalmente breve da enunciare ma per cui qualsiasi dimostrazione nel sistema formale F debba avere lunghezza superpolinomiale nella lunghezza dell'enunciato?

I risultati principali, dovuti a Cook, Reckhow, Krajíček, Ben-Sasson, Wigderson e molti altri, stabiliscono lower bound sulla lunghezza delle dimostrazioni in sistemi formali specifici — risoluzione proporzionale, Frege, cutting planes, Nullstellensatz, e altri. Questi lower bound sono risultati matematici rigorosi: non asseriscono che trovare la dimostrazione sia difficile, ma che qualsiasi dimostrazione nel sistema, una volta trovata, è necessariamente lunga.

Il collegamento con la monografia è questo: la lunghezza minima di una dimostrazione di P nel sistema F è esattamente la profondità logica D della comprensione che P richiede, nel sistema di rappresentazione corrispondente a F. Un lower bound sulla lunghezza delle dimostrazioni è dunque un lower bound sulla profondità cognitiva necessaria per comprendere P — non per verificare che P sia vera, ma per possedere una comprensione di P che ne renda la verità *perspicua*, nel senso che la dimostrazione porta inscritta.

VIII.2 Sistemi di dimostrazione come macchine di rappresentazione

Si formalizzi il collegamento. Un sistema di dimostrazione nel senso di Cook e Reckhow è una coppia (F, V) dove F è un insieme di assiomi e regole di inferenza, e V è un verificatore polinomialmente efficiente: dato un enunciato P e una stringa π, V(P, π) = 1 se e solo se π è una dimostrazione valida di P in F, e V è calcolabile in tempo polinomiale nella lunghezza di π e P.

La complessità di dimostrazione di P in F è:

$$pf_F(P) = \min\{ |\pi| : V(P, \pi) = 1 \}$$

il minimo è preso su tutte le dimostrazioni valide di P. Se P non è dimostrabile in F, si pone $pf_F(P) = \infty$.

Il collegamento con KC è dato dal seguente osservazione. Una dimostrazione π di P è un programma — una sequenza di passi verificabili — che produce P come output. La macchina che esegue questo programma è il verificatore V, interpretato come macchina di riscrittura: V legge le regole di inferenza e le applica agli assiomi per derivare P. La complessità di dimostrazione pf_F(P) è dunque la lunghezza del programma più breve per P nella macchina M_F corrispondente al sistema formale F.

In questa lettura:

$$pf_F(P) = K_\{M_F\}(P)$$

a meno di una costante additiva dipendente da F. La complessità di dimostrazione di P nel sistema F è la complessità di Kolmogorov di P rispetto alla macchina di rappresentazione M_F indotta da F. I lower bound sulla proof complexity sono lower bound su KC nel sistema di rappresentazione corrispondente.

VIII.3 Lower bound sulla profondità cognitiva: il collegamento formale

Il passaggio cruciale è estendere il collegamento dalla lunghezza delle dimostrazioni alla loro profondità computazionale. Si definisce la *profondità di dimostrazione* di P in F come:

$$pdepth_F(P) = \min\{ D_s(\pi) : V(P, \pi) = 1, |\pi| \leq pf_F(P) + s \}$$

il minimo della profondità logica tra tutte le dimostrazioni quasi-ottimali in lunghezza. Questa quantità misura quanto calcolo irriducibile è necessario per produrre la dimostrazione più concisa di P — non semplicemente quanto sia lunga, ma quanto processo computazionale essa porti inscritto.

La tesi che si sostiene è che pdepth_F(P) è un lower bound sulla profondità logica D_s della rappresentazione cognitiva che un agente deve possedere per *capire* P nel sistema di rappresentazione corrispondente a F. La giustificazione è la seguente: capire P significa disporre di una struttura cognitiva che rende perspicua la verità di P — che la genera, in senso computazionale, a partire da principi più fondamentali. Questa struttura è, per definizione, una dimostrazione di P (forse implicita, non formalmente

scritta, ma derivabile in linea di principio). La profondità di questa dimostrazione è il costo computazionale irriducibile della comprensione.

Si ha dunque:

$D_s(\text{comprensione di } P) \geq \text{pdepth}_F(P)$

I lower bound della proof complexity diventano lower bound cognitivi: dimostrano non solo che le dimostrazioni devono essere lunghe, ma che la comprensione deve essere profonda.

VIII.4 Il lower bound di Ben-Sasson e Wigderson per la risoluzione

Il risultato di proof complexity più direttamente interpretabile cognitivamente è il lower bound sulla *larghezza* delle dimostrazioni per risoluzione, dimostrato da Ben-Sasson e Wigderson nel 2001. Prima di enunciarlo, si introduce il contesto.

Il sistema di risoluzione opera su formule in forma normale congiuntiva — congiunzioni di clausole, dove ogni clausola è una disgiunzione di letterali. La regola di inferenza fondamentale è la risoluzione: da due clausole $C_1 \vee x$ e $C_2 \vee \neg x$ si deriva la clausola risolvente $C_1 \vee C_2$, eliminando il letterale x. Una dimostrazione di refutazione di una formula $F = C_1 \wedge \ldots \wedge C$ è una sequenza di clausole che inizia con le clausole di F e termina con la clausola vuota $\bot$, applicando la regola di risoluzione a ogni passo.

La *larghezza* di una dimostrazione è il numero massimo di letterali in una singola clausola della dimostrazione. Il risultato di Ben-Sasson e Wigderson stabilisce:

$$\text{Size}(F \vdash \bot) \geq 2^{\{W(F \vdash \bot) - W(F)\}}$$

dove Size è la lunghezza della dimostrazione (numero di clausole), $W(F \vdash \bot)$ è la larghezza minima necessaria per refutare F, e $W(F)$ è la larghezza iniziale delle clausole di F. Il lower bound è esponenziale nella differenza di larghezza: se refutare F richiede clausole significativamente più larghe delle clausole iniziali, la dimostrazione deve essere esponenzialmente lunga.

La traduzione cognitiva è immediata. La larghezza di una clausola nella dimostrazione misura quante proposizioni devono essere tenute in mente simultaneamente a un certo passo dell'inferenza — è una misura della *capacità di lavoro cognitivo* necessaria. Il lower bound di Ben-Sasson–Wigderson afferma che, se capire P richiede tenere in mente simultaneamente $W(F \vdash \bot)$ proposizioni, e le premesse ne contengono solo $W(F)$, allora la dimostrazione — la catena di inferenza che porta alle premesse alla conclusione — deve avere almeno $2^{\{W(F \vdash \bot) - W(F)\}}$ passi. Il costo cognitivo cresce esponenzialmente con la quantità di integrazione concettuale richiesta.

VIII.5 Il lower bound di Krajíček per i sistemi di Frege

I sistemi di Frege — sistemi di dimostrazione in logica proposizionale con assiomi e regole di inferenza finitamente molti, dove ogni passo deriva una formula da formule precedenti tramite regole di sostituzione — sono il sistema più vicino al ragionamento umano ordinario. Le dimostrazioni in un sistema di Frege corrispondono, approssimativamente, alle dimostrazioni "naturali" che un matematico scriverebbe su carta.

Krajíček ha dimostrato lower bound sulla lunghezza delle dimostrazioni in sistemi di Frege per specifiche tautologie — tra cui le tautologie di pigeonhole PHP_n^{n+1} (che asseriscono l'impossibilità di iniettare n+1 piccioni in n buche) e le tautologie di ordinamento. Per le tautologie di pigeonhole, qualsiasi dimostrazione in un sistema di Frege deve avere lunghezza almeno n^{Ω(log n)} — superpolinomiale in n.

Il significato cognitivo di questo risultato è il seguente. Il principio del pigeonhole è *ovvio* per un umano: non si possono mettere 5 piccioni in 4 buche con uno per buca. Ma questa ovvietà è ingannevole: nel sistema di rappresentazione della logica proposizionale senza estensioni, la dimostrazione del principio del pigeonhole per n generico è superpolinomialmente lunga. L'ovvietà soggettiva non riflette la brevità della dimostrazione nel sistema formale; riflette invece il fatto che il sistema di rappresentazione cognitivo umano dispone di primitive — la nozione di cardinalità, la comparazione diretta di insiemi finiti — che non sono disponibili nella logica proposizionale standard.

In termini di gap di rappresentazione (Capitolo VI): il gap tra il sistema di rappresentazione cognitivo umano M_human e il sistema di Frege M_Frege per la tautologia di pigeonhole è enorme. Un umano dimostra PHP_n^{n+1} in O(1) passi con la primitiva "confronta cardinalità"; il sistema di Frege richiede n^{Ω(log n)} passi perché quella primitiva non è disponibile.

VIII.6 Proof complexity e il teorema di incompletezza come geometria cognitiva

I lower bound della proof complexity convergono, dal basso, verso i teoremi di incompletezza di Gödel. Il risultato di Gödel — che ogni sistema formale sufficientemente potente contiene enunciati veri ma indimostrabi — può essere riformulato nel linguaggio della proof complexity: esistono tautologie per cui pf_F(P) = ∞, ossia che non hanno dimostrazione finita nel sistema F.

Il collegamento con il limite cognitivo discusso nel Capitolo IV è ora geometricamente preciso. I teoremi di incompletezza identificano enunciati con pdepth_F = ∞: la comprensione di quegli enunciati nel sistema di rappresentazione F richiede profondità infinita — è, per definizione, fuori dalla portata di qualsiasi agente che operi in F. Ma i lower bound della proof complexity identificano qualcosa di più sottile: enunciati con pdepth_F finita ma superpolinomiale, che sono in principio

comprensibili ma richiedono un tempo di elaborazione crescente con la dimensione del problema.

Si introduce la distinzione tra tre regioni dello spazio degli enunciati rispetto a un sistema formale F:

La prima è quella degli enunciati con pf_F(P) polinomiale in |P|: enunciati la cui dimostrazione è breve e la cui comprensione è computazionalmente efficiente. A questa regione appartengono la maggior parte dei risultati matematici elementari e i calcoli routinari.

La seconda è quella degli enunciati con pf_F(P) superpolinomiale ma finita: enunciati dimostrabili ma con dimostrazioni necessariamente lunghe nel sistema F. La comprensione di questi enunciati in F richiede profondità logica alta — è cognitivamente costosa nel senso preciso del triangolo. A questa regione appartengono, plausibilmente, molti risultati matematici profondi: teoremi la cui dimostrazione occupa decine o centinaia di pagine, e che non sono stati accorciati nonostante decenni di lavoro.

La terza è quella degli enunciati con pf_F(P) = ∞: enunciati indecidibili in F. La comprensione di questi enunciati in F è impossibile per qualsiasi agente con risorse finite che operi in F.

VIII.7 Il ruolo delle estensioni del sistema: lemmi come riduttori di profondità

Un meccanismo che la proof complexity ha studiato sistematicamente è quello dei *lemmi*: enunciati intermedi dimostrati separatamente e poi usati come assiomi aggiuntivi nelle dimostrazioni successive. L'aggiunta di un lemma L come assioma estende il sistema F a F' = F + {L}, e può ridurre drasticamente la lunghezza delle dimostrazioni di enunciati che dipendono da L.

Formalmente, il guadagno di lunghezza prodotto dall'aggiunta del lemma L per la dimostrazione di P è:

$$\Delta\text{Size}(L; F, P) = \text{pf}_F(P) - \text{pf}_{F+\{L\}}(P)$$

e il guadagno di profondità è:

$$\Delta\text{Depth}(L; F, P) = \text{pdepth}_F(P) - \text{pdepth}_{F+\{L\}}(P)$$

Il lemma è *cognitivamente vantaggioso* se il costo di comprenderlo — pdepth_F(L) — è inferiore al guadagno di profondità che produce:

$$\text{pdepth}_F(L) < \Delta\text{Depth}(L; F, P)$$

Quando questa condizione è soddisfatta, imparare il lemma prima di affrontare P riduce il costo cognitivo totale della comprensione. Quando non è soddisfatta, il lemma è cognitivamente controproducente: comprendere il lemma costa di più del guadagno che produce.

Questa è la formalizzazione della struttura curricolare ottimale: l'ordine in cui i concetti andrebbero introdotti è determinato dalla catena di disuguaglianze tra costi e guadagni di profondità. Un curriculum ottimale introduce i concetti nell'ordine in cui ogni concetto è un lemma cognitivamente vantaggioso per i concetti successivi — un ordine che non è in generale quello storico della scoperta, né quello logico dell'implicazione, ma quello che minimizza la profondità cognitiva totale richiesta per padroneggiare il dominio.

VIII.8 La congettura P vs NP come lower bound cognitivo universale

Il problema aperto più famoso della teoria della complessità — la congettura P ≠ NP — acquista nel quadro della monografia una lettura cognitiva che va oltre la sua formulazione standard.

La congettura P ≠ NP afferma che esistono problemi per cui verificare una soluzione è computazionalmente più efficiente che trovarla: esistono problemi in NP che non sono in P. Nel linguaggio della proof complexity, questo equivale — congetturalmente — all'esistenza di sistemi formali per cui non esiste un sistema di dimostrazione *polinomialmente simulato* da qualsiasi altro: sistemi per cui pf_{F'}(P) non è polinomialmente bounded da pf_F(P) per qualsiasi F'.

La traduzione cognitiva è questa: se P ≠ NP, allora esistono domini di comprensione per cui non esiste un sistema di rappresentazione M che riduca la profondità cognitiva necessaria a una funzione polinomiale della "dimensione" del dominio. Per tali domini, aggiungere primitive al proprio sistema di rappresentazione può ridurre il costo, ma non può eliminare la crescita superpolinomiale della profondità con la complessità dei problemi.

In positivo: la congettura P ≠ NP implica che esistono problemi per cui nessun artefatto cognitivo, nessun cambio di rappresentazione, nessuna collaborazione tra agenti può rendere la comprensione computazionalmente efficiente. La difficoltà non è accidentale — non dipende dalla scelta del sistema di rappresentazione sbagliato — ma è strutturale e irriducibile. Se P ≠ NP, il pensiero come compressione trova il suo limite più duro non nei teoremi di incompletezza ma nella geometria della complessità computazionale.

Se invece P = NP — la possibilità che la comunità della teoria della complessità considera altamente improbabile ma che non può escludere — allora esisterebbe un sistema di rappresentazione M universalmente efficiente in cui qualsiasi comprensione dimostrabile sarebbe raggiungibile in tempo polinomiale. Il pensiero come

compressione sarebbe, in quel mondo, senza limiti interni: ogni ostacolo sarebbe superabile con la scelta del sistema di rappresentazione giusto. Il pensiero come compressione sarebbe non solo una metafora della cognizione ma la sua legge fondamentale senza eccezioni strutturali. Che questo scenario sia quasi certamente falso è, forse, la migliore evidenza disponibile che la comprensione ha una dignità che non si riduce all'efficienza computazionale.

Appendici tecniche

Appendice A — Dimostrazione del teorema di invarianza di Kolmogorov

A.1 Enunciato

Siano U e V due macchine di Turing universali. Allora esiste una costante $c_{U,V} > 0$ tale che per ogni stringa $x \in \{0,1\}^*$:

$$| K_U(x) - K_V(x) | \leq c_{U,V}$$

La costante $c_{U,V}$ dipende dalla coppia (U, V) ma non dalla stringa x.

A.2 Dimostrazione

Per la definizione di universalità, V è in grado di simulare U: esiste un programma di simulazione s_{UV} tale che per ogni programma p, $V(s_{UV} \oplus p) = U(p)$, dove $\oplus$ denota la concatenazione con un opportuno separatore self-delimitante. La lunghezza $|s_{UV}|$ è una costante che dipende dalla coppia (U, V) ma non da p.

Sia p^* il programma ottimale per x rispetto a U: $U(p^*) = x$ e $|p^*| = K_U(x)$. Allora il programma $s_{UV} \oplus p^*$ è un programma valido per V che produce x:

$$V(s_{UV} \oplus p^*) = U(p^*) = x$$

e ha lunghezza $|s_{UV}| + |p^*| = c_{UV} + K_U(x)$, dove $c_{UV} = |s_{UV}|$. Per definizione di K_V come minimo:

$$K_V(x) \leq |s_{UV}| + K_U(x) = K_U(x) + c_{UV}$$

Per simmetria, scambiando i ruoli di U e V, si ottiene $K_U(x) \leq K_V(x) + c_{VU}$. Ponendo $c_{U,V} = \max(c_{UV}, c_{VU})$ si conclude:

$| K_U(x) - K_V(x) | \leq c_{U,V}$

A.3 Nota sulla self-delimitazione

La dimostrazione richiede che i programmi siano codificati in un modo *self-delimitante*: ogni programma deve indicare univocamente dove termina, senza che si debba conoscere a priori la sua lunghezza. Il modo standard è adoperare un prefisso binario auto-terminante, dove la lunghezza del programma è codificata in unario nel prefisso. Questa scelta assicura che la misura K sia *sub-additiva* nel senso che:

$K(x, y) \leq K(x) + K(y) + O(\log \min(K(x), K(y)))$

dove K(x, y) è la complessità della coppia. La self-delimitazione è necessaria per evitare ambiguità nella concatenazione $s_{UV} \oplus p^*$.

Appendice B — Dimostrazione della superficialità delle stringhe casuali

B.1 Enunciato

Sia x una stringa di lunghezza n con $K(x) \geq n - c$ per una costante c piccola (stringa c-incomprimibile). Allora la profondità logica $D_s(x)$ è al più polinomiale in n per qualsiasi s fissato.

B.2 Dimostrazione

Sia p^* il programma ottimale per x: $U(p^*) = x$ e $|p^*| \leq K(x) + s \leq n - c + s$. Poiché x è c-incomprimibile, $K(x) \geq n - c$, e quindi $|p^*| \geq n - c$ (nessun programma più corto può produrre x). Il programma p^* deve dunque contenere, in qualche forma codificata, almeno n - c bit di informazione relativa a x.

Il programma più naturale che soddisfa questo vincolo è quello che essenzialmente copia x: legge x come dato hardcodato nel proprio corpo e lo scrive sull'output. Un tale programma ha lunghezza $|p| = n + O(1)$ (i bit di x più un'intestazione di lunghezza costante) e si esegue in tempo $T(p) = O(n)$: deve scrivere n simboli sull'output, un simbolo per passo.

Per la definizione di D_s con $s = O(1)$, la profondità è il minimo tempo di esecuzione tra i programmi di lunghezza al più $K(x) + s \approx n + O(1)$. Il programma "copia" ha lunghezza $n + O(1)$ e tempo $O(n)$, ed è un candidato ammissibile. Dunque:

$D_s(x) \leq T(\text{"copia"}) = O(n)$

la profondità logica di una stringa incomprimibile è al più lineare nella sua lunghezza. ∎

B.3 Corollario

Il caos è computazionalmente pianeggiante: oggetti massimamente incomprimibili hanno profondità al più polinomiale. Alta KC e alta D sono condizioni mutuamente escludenti (a meno di costanti). La profondità alta è esclusivo appannaggio degli oggetti a bassa KC — degli oggetti con struttura regolare che richiedono un lungo processo computazionale per essere generati.

Appendice C — La disuguaglianza di Kraft e la probabilità algoritmica

C.1 Il codice prefisso e la disuguaglianza di Kraft

Un codice self-delimitante assegna a ogni programma p una lunghezza |p| tale che l'insieme dei programmi formi un codice prefisso: nessun programma è prefisso di un altro. La disuguaglianza di Kraft afferma che per qualsiasi codice prefisso su un alfabeto binario:

$$\Sigma_p \, 2^{-|p|} \leq 1$$

La somma è presa su tutti i programmi p nell'insieme di codice. La disuguaglianza garantisce che la misura di probabilità algoritmica:

$$m(x) = \Sigma_{p : U(p) = x} \, 2^{-|p|}$$

sia ben definita (la somma converge per ogni x, e la somma su tutti gli x è ≤ 1).

C.2 La misura di probabilità algoritmica di Solomonoff-Levin

La misura m(x) è la *probabilità algoritmica* di x: la probabilità che una macchina di Turing universale U, alimentata con un nastro di input casuale (ogni bit estratto indipendentemente con probabilità 1/2), produca x come output. Il termine $-\log_2 m(x)$ è la *complessità prefix-free* di x, spesso indicata K(x) nella letteratura moderna (in alcune trattazioni si distingue tra K e C per le versioni prefix-free e plain).

La relazione fondamentale tra m(x) e la complessità prefix-free è:

$$-\log_2 m(x) = K(x) + O(1)$$

a meno di costanti additive. Questo significa che la probabilità algoritmica di x è essenzialmente 2^{-K(x)}: gli oggetti con descrizione breve hanno probabilità alta, gli oggetti con descrizione lunga hanno probabilità bassa. La misura m è la formalizzazione del rasoio di Occam: assegna probabilità inversamente esponenziale alla lunghezza della descrizione.

C.3 La universalità di Kt nel senso di Levin

La complessità Kt di Levin è legata alla misura m attraverso la seguente identità (a costanti additive):

$$Kt(x) = -\log_2 m_t(x) + O(1)$$

dove m_t è la versione time-bounded della misura algoritmica:

$$m_t(x) = \Sigma_{p : U(p)=x,\ t(p) \leq t}\ 2^{-|p|}$$

La somma è ristretta ai programmi che producono x entro t passi. Al crescere di t, $m_t(x)$ cresce monotonicamente fino a $m(x)$, e $Kt(x)$ decresce fino a $K(x)$. La complessità Kt bilancia il contributo di tutti i programmi rapidi, pesandoli esponenzialmente per la lunghezza e non escludendo quelli subottimali: è una misura *bayesiana* sull'insieme dei programmi rapidi.

Appendice D — La ricerca universale di Levin e il suo costo

D.1 L'algoritmo di ricerca universale (Levin Search)

Il problema della ricerca universale è il seguente: dato un predicato verificabile P(x, y) — dove x è il problema e y è la soluzione candidata — trovare una y tale che P(x, y) = vero, oppure determinare che nessuna soluzione esiste.

L'algoritmo di Levin (1973) procede così. Si enumerano tutti i programmi $p_1, p_2, p_3, \ldots$ in ordine crescente di lunghezza, e si eseguono in modo interleaved: al passo k, si eseguono i programmi $p_1, \ldots, p_k$ ciascuno per $2^{k - |p_i|}$ passi aggiuntivi. Si verifica se l'output corrente di qualche programma costituisce una soluzione valida.

Il tempo totale impiegato dall'algoritmo per trovare una soluzione y^* di lunghezza $K(y^* \mid x)$ è:

$$T_{Levin} \leq c \cdot 2^{Kt(y^*|x)}$$

dove c è una costante che dipende da U ma non dal problema, e Kt(y* | x) è la complessità di Levin condizionata di y* dato x. L'algoritmo è *asintoticamente ottimale*: nessun altro algoritmo di ricerca può trovare la soluzione in tempo asintoticamente inferiore su tutti i problemi, a meno di un fattore costante dipendente dall'algoritmo.

D.2 Il fattore esponenziale e la sua inevitabilità

Il fattore 2^{Kt} è esponenziale in Kt(y* | x). Questo non è un difetto dell'algoritmo di Levin: è una conseguenza dell'indecidibilità del problema dell'arresto. Un qualsiasi algoritmo che trovi la soluzione ottimale su tutti i problemi deve avere un overhead almeno polinomiale rispetto all'algoritmo specifico per il problema, e in assenza di informazione sul problema, l'overhead esponenziale è inevitabile.

In pratica, l'algoritmo di Levin è inutilizzabile per problemi con Kt moderata: per Kt = 60 bit, il fattore 2^{60} ≈ 10^{18} rende il tempo di ricerca astronomico. Varianti più pratiche — come il metodo Hutter (AIXI) e le sue approssimazioni — cercano di sfruttare informazione a priori sul dominio per ridurre l'overhead, con risultati teoricamente eleganti ma computazionalmente ancora poco praticabili.

Appendice E — Il teorema di incompletezza e la sua relazione con KC

E.1 La connessione Gödel-Chaitin

Gregory Chaitin ha dimostrato che il teorema di incompletezza di Gödel è strettamente connesso all'incomprimibilità nel senso di KC. Il risultato principale, spesso chiamato *teorema di Chaitin*, è il seguente:

Sia F un sistema formale sufficientemente potente (che contiene l'aritmetica di Peano o equivalente). Esiste una costante L_F, che dipende da F ma non da x, tale che F non può dimostrare nessuna affermazione della forma "K(x) > c" per c > L_F.

In prosa: esistono stringhe x che sono genuinamente incomprimibili (K(x) > L_F + 1) ma di cui F non può dimostrare l'incomprimibilità. Il sistema formale ha un *limite di complessità dimostrabile*: non riesce a dimostrare che certi oggetti non hanno descrizione breve, anche quando effettivamente non ce l'hanno.

E.2 Derivazione informale

La costante L_F può essere stimata come segue. Sia |F| la complessità di Kolmogorov del sistema formale F stesso — la lunghezza del programma più breve

che genera tutte le teoremi di F. Allora L_F = O(|F|): il limite di complessità dimostrabile è dell'ordine della complessità del sistema formale stesso.

L'argomento informale è di tipo Berry-paradosso. Se F potesse dimostrare "K(x) > L_F" per qualche x, si potrebbe costruire un programma che cerca il primo x tale che F dimostra "K(x) > L_F" e poi lo stampa. Questo programma ha lunghezza O(|F|) — essenzialmente la lunghezza di F più un'intestazione costante — e produce un x con K(x) > L_F. Ma un programma di lunghezza O(|F|) non può produrre un output con K superiore alla propria lunghezza (più una costante), contraddicendo K(x) > L_F per L_F sufficientemente grande rispetto a O(|F|).

E.3 Implicazione per la cognizione

Il teorema di Chaitin implica che un agente cognitivo con un sistema di rappresentazione equivalente a F non può dimostrare che la propria comprensione di un dominio è ottimale — che la propria KC del dominio è minima — se la KC ottimale è superiore a L_F. L'agente può avere una comprensione eccellente (bassa KC) e non avere modo di *dimostrare* che non esiste una comprensione migliore. Questo connette il limite di incompletezza formale con il limite metacognitivo discusso nel Capitolo IV.

Appendice F — Note sulle misure di complessità per sequenze infinite

F.1 Complessità e normalità

Per stringhe finite la KC è ben definita; per sequenze infinite si introduce la nozione di *complessità* in diversi sensi. Una sequenza $x_1x_2x_3...$ è *Martin-Löf casuale* se le sue sezioni iniziali $x_1...x_n$ sono incomprimibili per ogni n a meno di una costante: $K(x_1...x_n) \geq n - c$ per qualche c fissato e ogni n. Le sequenze Martin-Löf casuali sono quelle che passano tutti i test statistici computabili: sono il corrispettivo algoritmico della casualità probabilistica.

La *normalità* è una nozione più debole: una sequenza è normale in base b se ogni blocco finito di cifre in base b appare con la frequenza limite prevista dalla distribuzione uniforme. Normalità non implica casualità Martin-Löf: esistono sequenze normali con struttura computazionale regolare (la concatenazione di tutti gli interi in ordine è normale ma non casuale nel senso di Martin-Löf). L'incomprimibilità è dunque più forte della normalità e cattura meglio l'assenza di struttura.

F.2 Dimensione di Hausdorff algoritmica

Per sequenze infinite si definisce anche la *dimensione algoritmica di Hausdorff*, che misura la densità di informazione asintotica:

$$\dim(x) = \lim\inf_{n\to\infty} \; K(x_1 \ldots x) \,/\, n$$

Una sequenza casuale nel senso di Martin-Löf ha dim = 1; una sequenza periodica ha dim = 0; una sequenza che porta, per esempio, la struttura di un frattale autosimile ha dim $\in (0,1)$ pari alla dimensione frattale della struttura. La dimensione algoritmica di Hausdorff generalizza in modo naturale la dimensione geometrica all'informazione.

Il pensiero come compressione

Appendici tecniche — Secondo gruppo (Capitoli VI, VII, VIII)

Appendice G — Il gap di simulazione per macchine di rappresentazione parziali

G.1 Macchine di rappresentazione: definizione formale

Si definisce una *macchina di rappresentazione* M come una macchina di Turing con un insieme finito di primitive $P = \{f_1, \ldots, f\}$, dove ogni f_i è un'operazione elementare con costo unitario. M differisce da una macchina di Turing universale in quanto P può essere strettamente contenuto nell'insieme di tutte le operazioni computazionali: M non è necessariamente universale.

Per ogni macchina di rappresentazione M si definisce la *complessità di Kolmogorov rispetto a M* come:

$$K_M(x) = \min\{\, |p| : M(p) = x \,\}$$

dove il minimo è preso su tutti i programmi p — sequenze di operazioni in P — che producono x come output. Se nessun programma in M produce x, si pone $K_M(x) = \infty$.

G.2 Enunciato del teorema sul gap di simulazione

Siano M_α e M_β due macchine di rappresentazione con insiemi di primitive P_α e P_β. Si dice che M_α *simula* M_β a costo $f(n)$ se esiste un programma di simulazione $s_{\alpha\beta} \in P_\alpha^*$ tale che per ogni programma $p \in P_\beta^*$ con $|p| = n$:

$$M_\alpha(s_{\alpha\beta} \oplus p) = M_\beta(p) \quad e \quad |s_{\alpha\beta} \oplus p| \leq |p| + f(n)$$

Per macchine universali, $f(n) = O(1)$ — il teorema di invarianza standard. Per macchine di rappresentazione parziali, $f(n)$ può essere non costante.

Teorema (Gap di simulazione). Sia $f_\alpha \in P_\beta \setminus P_\alpha$ una primitiva disponibile a β ma non a α. Sia x_n la famiglia di stringhe tale che il programma ottimale in M_β per x_n usa f_α esattamente n volte, e ogni sostituzione di f_α con programmi in P_α richiede un programma di lunghezza almeno $g(n)$ in M_α. Allora:

$$K_{M_\alpha}(x_n) - K_{M_\beta}(x_n) \geq n \cdot g(n) - O(\log n)$$

Il gap di simulazione cresce almeno linearmente nel numero di applicazioni della primitiva mancante, moltiplicato per il costo di simularla.

G.3 Dimostrazione

Sia p_β il programma ottimale per x_n in M_β: $M_\beta(p_\beta) = x_n$ e $|p_\beta| = K_{M_\beta}(x_n)$. Il programma p_β usa f_α esattamente n volte per ipotesi. Ogni occorrenza di f_α in p_β deve essere sostituita, nel programma corrispondente in M_α, con una subroutine s_i di lunghezza almeno $g(n)$. La sostituzione produce un programma p_α in M_α con:

$$|p_\alpha| \geq |p_\beta| - n \cdot |f_\alpha| + n \cdot g(n)$$

dove $|f_\alpha| = O(1)$ è il costo sintattico di un'occorrenza di f_α nel programma originale. Poiché p_α è un candidato per il minimo su cui è definita $K_{M_\alpha}(x_n)$:

$$K_{M_\alpha}(x_n) \leq |p_\alpha| \leq K_{M_\beta}(x_n) - n \cdot O(1) + n \cdot g(n) + O(\log n)$$

dove il termine $O(\log n)$ proviene dall'overhead di indicizzazione delle subroutine. Questo fornisce un upper bound. Per il lower bound, si osserva che qualsiasi programma p in M_α che produca x_n deve "simulare" gli n usi di f_α, e ciascuno richiede almeno $g(n)$ operazioni per ipotesi. Dunque:

$$K_{M_\alpha}(x_n) \geq K_{M_\beta}(x_n) + n \cdot g(n) - O(\log n)$$

Combinando upper e lower bound:

$$K_{M_\alpha}(x_n) - K_{M_\beta}(x_n) = \Theta(n \cdot g(n))$$

Il gap cresce come $n \cdot g(n)$, che è non costante quando $g(n) \geq 1$ per ogni n. ∎

G.4 Corollario: non-boundedness del gap per famiglie profonde

Se x_n è una famiglia tale che $g(n)$ cresce con n — per esempio $g(n) = \Omega(\log n)$ — allora il gap di simulazione $K_{M_\alpha}(x_n) - K_{M_\beta}(x_n)$ cresce senza limite. Il teorema di invarianza di Kolmogorov non si applica a macchine di rappresentazione parziali per famiglie di oggetti la cui complessità cresce con la profondità delle primitive mancanti.

Questo corollario è il risultato formale sottostante all'affermazione del Capitolo VI che l'incomprensione strutturale tra agenti con sistemi di rappresentazione diversi cresce arbitrariamente con la profondità degli oggetti trattati.

Appendice H — Dimostrazione del teorema di aumento della profondità per proiezione al modello

H.1 Notazione e premesse

Si usa la decomposizione ottimale nel senso della sofisticazione: per una stringa x con sofisticazione $\text{soph}_c(x)$, la coppia (p_x, d_x) soddisfa $U(p_x, d_x) = x$ e $K(x \mid p_x) \geq K(x) - c$. Si indica con $T^*(p_x)$ il tempo di esecuzione del programma ottimale per p_x, e con $T^*(x)$ il tempo di esecuzione del programma ottimale per x.

H.2 Enunciato preciso

Teorema. Sia (p_x, d_x) la decomposizione ottimale di x. Sia T_{apply} il tempo necessario per eseguire la computazione $U(p_x, d_x) = x$ una volta noti p_x e d_x — il tempo dell'*applicazione* del modello al residuo. Allora:

$$D_s(p_x) \geq D_s(x) - T_{\text{apply}} - O(|d_x|)$$

Corollario. Se $T_{\text{apply}} = O(|d_x|)$ — se il tempo di applicazione è lineare nella lunghezza del residuo — allora:

$$D_s(p_x) \geq D_s(x) - O(|d_x|)$$

H.3 Dimostrazione

Sia q* il programma quasi-ottimale per x che realizza D_s(x): U(q*) = x, |q*| ≤ K(x) + s, e t(q*) = D_s(x). Il programma q* produce x in tempo D_s(x).

Si costruisce ora un programma r per p_x nel modo seguente. Il programma r simula q* fino al momento in cui q* ha generato p_x come sottoprodotto intermedio del calcolo — o, se q* non genera p_x esplicitamente come intermedio, r esegue una ricerca sul nastro di q* per identificare p_x tra i simboli scritti. Il tempo necessario per questa identificazione è al più T*(q*) = D_s(x). Una volta identificato p_x, r lo scrive sull'output e si arresta.

La lunghezza di r è |q*| + O(1) ≤ K(x) + s + O(1): r è quasi-ottimale in lunghezza per p_x se K(p_x) ≤ K(x) (il modello è almeno altrettanto comprimibile della stringa completa, il che è garantito dalla definizione di sofisticazione con c ≥ 0). Il tempo di esecuzione di r è t(r) ≤ D_s(x) + O(1).

Per definizione di D_s(p_x) come minimo su programmi quasi-ottimali:

D_s(p_x) ≤ t(r) ≤ D_s(x) + O(1)

Questo è un upper bound. Per il lower bound, si considera il programma ottimale per p_x — sia esso π*, con t(π*) = D_s(p_x). Si costruisce da π* un programma per x: si esegue π* per ottenere p_x, poi si applica p_x a d_x per ottenere x. Il tempo totale è t(π*) + T_apply + O(|d_x|). Questo programma è un candidato per il minimo su cui è definita D_s(x):

D_s(x) ≤ D_s(p_x) + T_apply + O(|d_x|)

Riorganizzando:

D_s(p_x) ≥ D_s(x) - T_apply - O(|d_x|)

I

H.4 Il caso dell'acceleratore computazionale

Si dimostra ora quando D_s(p_x) > D_s(x) — quando la proiezione al modello *aumenta strettamente* la profondità.

Proposizione. d_x è un acceleratore computazionale di p_x se e solo se il programma ottimale per x usa d_x come *oracolo* — legge d_x per saltare passi computazionali che il programma ottimale per p_x deve invece percorrere per intero.

Formalmente, si dice che d_x è un acceleratore se esiste una fase del calcolo di q* in cui q* legge un simbolo di d_x e, sulla base di quel simbolo, salta a una posizione nel calcolo che il programma π^* per p_x raggiunge solo dopo Δ passi aggiuntivi, con $\Delta > |d_x|$. In questo caso:

$$D_s(x) < D_s(p_x) - \Delta + O(|d_x|)$$

ovvero:

$$D_s(p_x) > D_s(x) + \Delta - O(|d_x|) > D_s(x)$$

per $\Delta > O(|d_x|)$. La profondità del modello è strettamente maggiore di quella della stringa completa. ∎

Questo accade precisamente quando il residuo contiene informazione che abbrevia la ricerca nel calcolo del modello — quando d_x funziona come un *certificato* che guida il programma verso la soluzione senza percorrere l'intero spazio di ricerca.

Appendice I — Il lower bound di Ben-Sasson e Wigderson per la risoluzione

I.1 Il sistema di risoluzione

Il sistema di risoluzione opera su formule in forma normale congiuntiva $F = C_1 \wedge ... \wedge C$. Una *refutazione* di F è una sequenza $\pi = (D_1, D_2, ..., D)$ di clausole tale che $D_1 = \bot$ (clausola vuota) oppure ogni D_i è una clausola iniziale di F o si ottiene da due clausole precedenti D , D con j, k < i tramite la regola di risoluzione:

Da $(A \vee x)$ e $(B \vee \neg x)$ si deriva $(A \vee B)$

La *dimensione* della refutazione è Size(π) = t (numero di clausole). La *larghezza* della refutazione è W(π) = max_i |D_i| (numero massimo di letterali in una clausola). Si definisce:

W(F ⊢ ⊥) = min{ W(π) : π è una refutazione di F }

W(F) = max_i |C_i| (larghezza delle clausole iniziali)

I.2 Enunciato del lower bound

Teorema (Ben-Sasson, Wigderson 2001). Per ogni formula F insoddisfacibile:

Size(F ⊢ ⊥) ≥ 2^{ W(F ⊢ ⊥) - W(F) }

I.3 Schema della dimostrazione

La dimostrazione usa il metodo dei *giochi di Ehrenfeucht–Fraïssé* adattato alla risoluzione. Si definisce una sequenza di insiemi S_0, S_1, ..., S_W di clausole, costruita come segue. S_0 = {⊥} (la clausola vuota). Per ogni i, S_{i+1} è l'insieme di tutte le clausole che si ottengono da clausole in S_i applicando la regola di risoluzione *a ritroso* — cioè espandendo ⊥ aggiungendo letterali — fino a raggiungere clausole di larghezza i+1.

Si dimostra per induzione che ogni clausola in S_i ha larghezza al più i + W(F ⊢ ⊥) - W(F). Poiché ogni refutazione deve passare per ⊥ e risalire alle clausole iniziali, ogni passo della refutazione deve "attraversare" almeno uno degli insiemi S_i. Il numero di clausole in S_i cresce almeno come 2^i per un argomento di separazione — due clausole di S_i che differiscono in un letterale richiedono dimostrazioni separate. Il totale è:

Size(F ⊢ ⊥) ≥ |S_{W(F ⊢ ⊥) - W(F)}| ≥ 2^{ W(F ⊢ ⊥) - W(F) }

I

I.4 Applicazione alle tautologie di pigeonhole

La famiglia di formule PHP_n^{n+1} codifica il principio del pigeonhole: non esiste una funzione iniettiva da [n+1] a [n]. Le variabili p_{i,j} (con 1 ≤ i ≤ n+1, 1 ≤ j ≤ n) rappresentano "il piccione i occupa la buca j". Le clausole sono:

Clausole di copertura: per ogni piccione i, la clausola (p_{i,1} ∨ p_{i,2} ∨ ... ∨ p_{i,n}) — ogni piccione occupa almeno una buca.

Clausole di collisione: per ogni buca j e ogni coppia di piccioni i ≠ i', la clausola (¬p_{i,j} ∨ ¬p_{i',j}) — due piccioni non occupano la stessa buca.

Ben-Sasson e Wigderson dimostrano che W(PHP_n^{n+1} ⊢ ⊥) = n + 2 mentre W(PHP_n^{n+1}) = n. Dunque:

Size(PHP_n^{n+1} ⊢ ⊥) ≥ 2^{(n+2) - n} = 4

Questo lower bound è debole; lower bound esponenziali più forti per la risoluzione su PHP sono noti attraverso tecniche diverse, in particolare il lower bound di Haken del 1985 che stabilisce:

Size(PHP_n^{n+1} ⊢ ⊥) ≥ 2^{n/20}

per la risoluzione, confermando che qualsiasi dimostrazione per risoluzione del pigeonhole richiede lunghezza esponenziale in n.

Appendice L — Lower bound di Krajíček per sistemi di Frege

L.1 Sistemi di Frege: definizione

Un *sistema di Frege* F è un sistema di dimostrazione per la logica proposizionale con le seguenti proprietà. L'insieme di assiomi è un insieme finito di schemi di tautologie. Le regole di inferenza sono un insieme finito di regole che permettono di derivare nuove formule da formule precedenti tramite sostituzione uniforme di variabili proposizionali con formule arbitrarie. La regola di modus ponens (da A e A → B si deriva B) è sempre inclusa.

La lunghezza di una dimostrazione è il numero di formule nella sequenza di derivazione. La complessità di Frege di una tautologia P è:

pf_F(P) = min{ |π| : π è una dimostrazione di P in F }

L.2 Il lower bound per il pigeonhole in sistemi di Frege

Teorema (Krajíček, Pudlák, Woods, 1992). Per ogni sistema di Frege F, la complessità delle tautologie di pigeonhole PHP_n^{n+1} soddisfa:

$pf_F(PHP_n^{n+1}) \geq n^{\Omega(\log n / \log \log n)}$

Il lower bound è superpolinomiale in n. Non si conosce un lower bound esponenziale per sistemi di Frege — la questione è aperta e connessa alla congettura $P \neq NP$ — ma il lower bound superpolinomiale è sufficiente per mostrare che i sistemi di Frege non dimostrano il pigeonhole efficientemente.

L.3 Schema della dimostrazione

La dimostrazione usa la *complessità della comunicazione* come strumento. Si osserva che una dimostrazione corta di PHP_n^{n+1} in un sistema di Frege implicherebbe l'esistenza di un protocollo di comunicazione efficiente per un certo problema combinatoriale correlato al pigeonhole — problema che è noto richiedere comunicazione $\Omega(n)$ bit. Questo contraddice il lower bound di comunicazione.

Formalmente, si associa a ogni dimostrazione π di PHP_n^{n+1} un protocollo di comunicazione P_π tra due parti Alice e Bob, dove Alice conosce l'assegnamento delle righe e Bob quello delle colonne della matrice del pigeonhole. Si dimostra che la complessità di comunicazione di P_π è $\Omega(\log(|\pi|))$, e quindi che $|\pi| \geq 2^{\Omega(\text{comunicazione})} \geq 2^{\Omega(n)}$ — ma questo darebbe un lower bound esponenziale, che non si riesce a dimostrare con le tecniche note. Il lower bound superpolinomiale si ottiene con una versione più debole dell'argomento, che considera protocolli con numero limitato di round.

L.4 Il lower bound di Razborov per sistemi di Frege con conteggio

Razborov ha dimostrato lower bound esponenziali per sistemi di Frege estesi con assiomi di conteggio in caratteristica diversa dalla caratteristica del problema in esame. In particolare, sistemi di Frege con assiomi mod-2 non dimostrano efficientemente il principio del pigeonhole mod-3, e viceversa. Questo risultato è rilevante per la lettura cognitiva: le primitive aritmetiche disponibili al sistema di rappresentazione determinano quali lower bound si abbassano e quali rimangono superpolinomiali — la struttura modulare del conteggio è una primitiva la cui disponibilità riduce drasticamente la profondità cognitiva necessaria per certi domini e non per altri.

Bibliografia ragionata

La bibliografia che segue è organizzata per nuclei tematici corrispondenti ai movimenti della monografia. Per ciascun testo si fornisce una nota sul suo ruolo specifico

nell'argomentazione, con l'indicazione delle tesi o dei risultati rilevanti. Non si tratta di una rassegna esaustiva della letteratura: è una mappa dei testi che hanno nutrito o delimitato le posizioni sostenute.

I. Complessità di Kolmogorov, profondità logica e misure correlate

Kolmogorov, A. N. (1965). Three approaches to the quantitative definition of information. *Problems of Information Transmission*, 1(1), 1–7.

Il testo fondativo. Kolmogorov introduce tre definizioni della quantità di informazione contenuta in una stringa — combinatoria, probabilistica e algorítmica — e argomenta per la loro equivalenza asintotica. La terza definizione, quella che porta il suo nome, è il punto di partenza dell'intera monografia. Da notare: la versione del 1965 non tratta ancora la self-delimitazione; questa verrà introdotta da Levin e da Chaitin.

Solomonoff, R. J. (1964). A formal theory of inductive inference. *Information and Control*, 7(1), 1–22; 7(2), 224–254.

Solomonoff sviluppa indipendentemente la nozione di probabilità algoritmica e il principio di induzione basata sulla compressione. Il suo lavoro precede quello di Kolmogorov di un anno nella produzione, sebbene la pubblicazione sia quasi simultanea. Rilevante per le sezioni III.2 e l'Appendice C sulla misura di Solomonoff-Levin.

Levin, L. A. (1974). Laws of information conservation (nongrowth) and aspects of the foundation of probability theory. *Problems of Information Transmission*, 10(3), 206–210.

In questo testo Levin introduce la complessità Kt e la ricerca universale. Il risultato di universalità di Kt — la sua coincidenza con -log m a meno di costanti — è dimostrato qui. È uno dei risultati più sottili e meno citati della teoria della complessità algorítmica, e costituisce il secondo vertice del triangolo nella monografia.

Bennett, C. H. (1988). Logical depth and physical complexity. In R. Herken (Ed.), *The Universal Turing Machine: A Half-Century Survey* (pp. 227–257). Oxford University Press.

Il testo in cui Bennett introduce formalmente la profondità logica. La definizione, le tre classi (banale, casuale, profondo), il teorema sulla superficialità delle stringhe casuali e le relazioni con la seconda legge della termodinamica sono tutti in questo saggio. È il testo più importante per il Capitolo I della monografia, e il punto di partenza per tutto ciò che viene sviluppato oltre.

Li, M., & Vitányi, P. M. B. (2019). *An Introduction to Kolmogorov Complexity and Its Applications* (4ª ed.). Springer.

Il riferimento tecnico standard. Contiene dimostrazioni rigorose di tutti i risultati citati nella monografia — invarianza, incomprimibilità, probabilità algoritmica, teorema di Chaitin, dimensione algoritmica di Hausdorff — con un'attenzione pedagogica che non sacrifica il rigore. L'Appendice A della monografia segue la struttura della dimostrazione del teorema di invarianza presentata in questo testo.

Koppel, M. (1988). Complexity, depth, and sophistication. *Complex Systems*, 2(4), 1087–1091.

L'articolo in cui viene introdotta la sofisticazione. Breve ma denso: Koppel definisce la misura, dimostra che le stringhe casuali e quelle banali hanno entrambe sofisticazione bassa, e argomenta per l'identificazione della sofisticazione con la "complessità organizzata" nel senso biologico. Rilevante per l'intero Capitolo V.

Gell-Mann, M., & Lloyd, S. (1996). Information measures, effective complexity, and total information. *Complexity*, 2(1), 44–52.

Gell-Mann e Lloyd propongono la complessità effettiva come alternativa alla sofisticazione, partendo da motivazioni fisiche e biologiche anziché computazionali. Il confronto tra EC e soph discusso in V.4 è fondato su questo articolo. La nozione di "schema" come struttura regolare di un sistema complesso è qui sviluppata con chiarezza.

Chaitin, G. J. (1975). A theory of program size formally identical to information theory. *Journal of the ACM*, 22(3), 329–340.

Chaitin sviluppa la versione prefix-free della complessità di Kolmogorov e la connette all'incompletezza di Gödel. Il teorema discusso nell'Appendice E — il limite di complessità dimostrabile — è stabilito qui in forma tecnica precisa. Chaitin ha sviluppato questa connessione in modo più accessibile in testi successivi, ma questa è la fonte primaria.

II. Cognizione, insight e fenomenologia del comprendere

Ohlsson, S. (1992). Information-processing explanations of insight and related phenomena. In M. T. Keane & K. J. Gilhooly (Eds.), *Advances in the Psychology of Thinking* (Vol. 1, pp. 1–44). Harvester Wheatsheaf.

Ohlsson sviluppa la teoria dell'impasse come condizione necessaria precedente l'insight. Il concetto di rappresentazione che ha "esaurito lo spazio di ricerca" è qui formulato in modo sistematico e collegato alla letteratura sulla risoluzione di problemi. Rilevante per II.3 e il modello dell'impasse come minimo locale di Kt.

Metcalfe, J., & Wiebe, D. (1987). Intuition in insight and noninsight problem solving. *Memory & Cognition*, 15(3), 238–246.

Lo studio empirico che documenta la differenza fenomenologica tra risoluzione graduale e insight. Le curve di "warmth" — la sensazione di vicinanza alla soluzione — mostrano il profilo piatto-poi-verticale che il modello formale del Capitolo II cerca di spiegare. Un testo di psicologia sperimentale di qualità solida, raramente citato fuori dalla letteratura specialistica.

Köhler, W. (1925). *The Mentality of Apes* (E. Winter, Trans.). Harcourt, Brace.

Gli esperimenti con gli scimpanzé di Tenerife restano uno dei documenti più vividi del fenomeno dell'insight in condizioni osservabili. Sultan che impila le casse o connette i bastoni non sta eseguendo una ricerca graduale: sta effettuando una ristrutturazione improvvisa della situazione. Il riferimento è qui soprattutto per la solidità fenomenologica dell'osservazione originale.

Kris, E. (1952). *Psychoanalytic Explorations in Art*. International Universities Press.

Il concetto di "regressione al servizio dell'Io" — la capacità di accedere temporaneamente a modalità di pensiero meno organizzate come preludio alla creazione — è sviluppato qui nel contesto della psicologia dell'arte. Il collegamento con la riduzione temporanea di D discussa in II.5 è interpretativo, non diretto; ma l'osservazione clinica di Kris ha una struttura che il modello formale illumina retroattivamente.

Hutchins, E. (1995). *Cognition in the Wild*. MIT Press.

Il testo fondativo della cognizione distribuita. Hutchins studia il sistema di navigazione di una nave da guerra come sistema cognitivo distribuito tra agenti umani, strumenti e artefatti. La tesi — che l'unità di analisi cognitiva corretta sia il sistema, non il singolo agente — è qui argomentata con ricchezza etnografica e rigore teorico. Rilevante per la discussione degli artefatti come riduttori di barriera in II.7.

Kahneman, D. (2011). *Thinking, Fast and Slow*. Farrar, Straus and Giroux.

La distinzione sistema-1/sistema-2 è qui presentata nella forma più elaborata e accessibile. Il collegamento con la complessità resource-bounded (K^t) discusso in V.7 non è nel testo di Kahneman, che non usa strumenti formali della teoria della complessità: è un'elaborazione della monografia. Ma la fenomenologia descritta da Kahneman è la base empirica per quella connessione.

III. Teoria dell'informazione, Shannon e il contesto

Shannon, C. E. (1948). A mathematical theory of communication. *Bell System Technical Journal*, 27(3), 379–423; 27(4), 623–656.

Il testo fondativo della teoria dell'informazione. L'entropia di Shannon, i teoremi sulla codifica di sorgente e sul canale, la relazione tra compressione e codifica ottimale sono

qui dimostrati per la prima volta. Il presupposto della monografia — che KC e Shannon siano già noti — si fonda su questo testo come antefatto.

Cover, T. M., & Thomas, J. A. (2006). *Elements of Information Theory* (2ª ed.). Wiley-Interscience.

Il riferimento tecnico standard per la teoria dell'informazione. Contiene i risultati su cross-entropia, codifica, canale e relative dimostrazioni. Utile come sfondo per la sezione III.2, dove la cross-entropia dei Transformer è collegata alla compressione KC nel senso di Shannon.

IV. Modelli di linguaggio e architetture Transformer

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., & Polosukhin, I. (2017). Attention is all you need. *Advances in Neural Information Processing Systems*, 30.

L'articolo che introduce l'architettura Transformer. La struttura di self-attention — la formula $\text{Attention}(Q,K,V) = \text{softmax}(QK^T/\sqrt{d_k})\cdot V$ discussa in III.3 — è qui presentata. Il costo computazionale $O(n^2d)$ per strato è una conseguenza diretta di questa struttura.

Wei, J., Wang, X., Schuurmans, D., Bosma, M., Xia, F., Chi, E., Le, Q. V., & Zhou, D. (2022). Chain-of-thought prompting elicits reasoning in large language models. *Advances in Neural Information Processing Systems*, 35.

Lo studio sistematico che documenta il miglioramento prodotto dal chain-of-thought su compiti di ragionamento multi-step. I risultati empirici motivano l'analisi formale della sezione III.4: il chain-of-thought come meccanismo di aumento della profondità computazionale effettiva.

Hutter, M. (2004). *Universal Artificial Intelligence: Sequential Decisions Based on Algorithmic Probability*. Springer.

Hutter sviluppa AIXI — l'agente ideale che combina la ricerca universale di Levin con la teoria della decisione bayesiana. Il modello è computazionalmente intrattabile, ma fornisce il riferimento teorico rispetto al quale valutare i sistemi pratici. Rilevante per le Appendici C e D sulla ricerca universale e la misura algoritmica.

V. Biologia, complessità organizzata, autonomia

Atlan, H. (1979). *Entre le cristal et la fumée: Essai sur l'organisation du vivant*. Seuil.

Il testo più accessibile del programma di Atlan sulla complessità biologica. Il titolo cattura l'intuizione centrale: gli organismi viventi si collocano tra il cristallo (ordine totale, informazione zero) e il fumo (disordine totale, informazione massima). La nozione di organizzazione come riduzione della casualità attraverso l'interazione interna è qui sviluppata con chiarezza. Rilevante per V.5.

Koppel, M., & Atlan, H. (1991). An almost machine-independent theory of program-length complexity, sophistication, and induction. *Information Sciences*, 56(1–3), 23–33.

Il testo in cui Koppel e Atlan sviluppano congiuntamente la sofisticazione nel contesto biologico e ne dimostrano proprietà formali. Il risultato di quasi-indipendenza dalla macchina — la sofisticazione è stabile rispetto alla scelta di U a meno di costanti, come KC — è dimostrato qui.

VI. Filosofia della mente e hard problem

Chalmers, D. J. (1996). *The Conscious Mind: In Search of a Fundamental Theory*. Oxford University Press.

La formulazione più influente dell'hard problem of consciousness. La distinzione tra "easy problems" (spiegare le funzioni cognitive) e "hard problem" (spiegare perché esiste esperienza soggettiva) è qui argomentata con precisione filosofica. Il Capitolo IV della monografia usa questa distinzione come riferimento per il terzo limite del programma di compressione, senza prendere posizione sulla soluzione del problema.

Nagel, T. (1974). What is it like to be a bat? *Philosophical Review*, 83(4), 435–450.

Il saggio che ha reso celebre la nozione di prospettiva di prima persona come elemento irriducibile dell'esperienza. L'argomentazione di Nagel — che nessuna descrizione fisica terza-persona catturi "cosa significa" avere un'esperienza soggettiva — è il precedente diretto del limite fenomenologico discusso in IV.6. Il saggio è breve, denso e rimane uno dei documenti più leggibili della filosofia della mente del Novecento.

Searle, J. R. (1980). Minds, brains, and programs. *Behavioral and Brain Sciences*, 3(3), 417–424.

L'argomento della stanza cinese. Searle sostiene che la manipolazione sintattica di simboli — indipendentemente dalla complessità — non può produrre comprensione semantica. Il Capitolo III non assume la correttezza di Searle: il punto non è che i modelli di linguaggio non "capiscano" in senso assoluto, ma che realizzino un tipo specifico di compressione (bassa D) diverso da quello che caratterizza la comprensione umana in domini profondi. La distinzione è formale, non metafisica.

VII. Testi di sfondo e riferimenti incrociati

Hofstadter, D. R. (1979). *Gödel, Escher, Bach: An Eternal Golden Braid*. Basic Books.

Citato non per i risultati tecnici — che sono spesso presentati in modo informale — ma per la capacità di mostrare come i circoli autoreferenziali dell'incompletezza di Gödel, il contrappunto di Bach e il paradosso del sé si corrispondano strutturalmente. L'intuizione che la profondità e l'autoreferenza siano connesse — che sistemi capaci di rappresentare se stessi abbiano una complessità logica qualitativamente diversa — attraversa l'intero lavoro di Hofstadter e fornisce un contesto culturale ricco alla parte più formale della monografia.

Rissanen, J. (1978). Modeling by shortest data description. *Automatica*, 14(5), 465–471.

Rissanen introduce il *Minimum Description Length* (MDL) come principio di selezione del modello statistico: tra tutti i modelli compatibili con i dati, preferire quello che minimizza la somma della lunghezza del modello e della lunghezza dei dati compressi dal modello. Il MDL è la versione statistica pratica del principio KC, e il suo collegamento con la sofisticazione (che separa esattamente la lunghezza del modello da quella dei dati) è diretto. Non discusso esplicitamente nella monografia, ma sottostante a molti degli argomenti del Capitolo III.

Vapnik, V. N. (1998). *Statistical Learning Theory*. Wiley.

Il framework della dimensione VC e della generalizzazione è il contesto in cui la compressione — nel senso di fitting con modelli di complessità controllata — è stata più sistematicamente studiata in machine learning. Il collegamento con KC non è esplicito in Vapnik, ma la struttura formale è parallela: la dimensione VC misura la capacità descrittiva di una classe di funzioni in modo analogo a come KC misura la lunghezza della descrizione di una singola stringa.

VIII. Gap di rappresentazione, comunicazione e traduzione concettuale

Cook, S. A., & Reckhow, R. A. (1979). The relative efficiency of propositional proof systems. *Journal of Symbolic Logic*, 44(1), 36–50.

Il testo fondativo della proof complexity moderna. Cook e Reckhow definiscono formalmente i sistemi di dimostrazione come coppie (F, V) con verificatore polinomiale, e formulano la domanda centrale del campo: esiste un sistema di dimostrazione *ottimale*, tale che qualsiasi altra dimostrazione vi sia simulata con overhead polinomiale? La risposta affermativa equivale — come Cook e Reckhow dimostrano — alla tesi P = NP. La sezione VIII.2 della monografia riposa interamente su questo framework.

Krajíček, J. (1995). *Bounded Arithmetic, Propositional Logic, and Complexity Theory*. Cambridge University Press.

Il riferimento tecnico standard per i lower bound nei sistemi di Frege e nei sistemi di aritmetica bounded. Contiene la dimostrazione del lower bound superpolinomiale per il pigeonhole in sistemi di Frege (con Pudlák e Woods), sviluppata attraverso la complessità della comunicazione. L'Appendice L segue la struttura argomentativa di questo testo. Krajíček è anche l'autore di lavori più recenti sulla connessione tra proof complexity e teoria degli algoritmi che hanno alimentato la congettura TFNP e le sue relazioni con la complessità computazionale.

Ben-Sasson, E., & Wigderson, A. (2001). Short proofs are narrow — resolution made simple. *Journal of the ACM*, 48(2), 149–169.

Il testo da cui è tratto il lower bound Size $\geq 2^{W(F\vdash\bot) - W(F)}$ discusso nella sezione VIII.4 e dimostrato nell'Appendice I. Il titolo è programmatico: dimostrazioni brevi in risoluzione sono necessariamente strette — usano clausole di larghezza limitata — e questo ne limita la potenza espressiva. Il risultato ha avuto influenza notevole in proof complexity perché fornisce un metodo sistematico per derivare lower bound sulla dimensione a partire da lower bound sulla larghezza, spesso più facili da stabilire.

Razborov, A. A. (1998). Lower bounds for the polynomial calculus. *Computational Complexity*, 7(4), 291–324.

Razborov stabilisce lower bound per il sistema del *polynomial calculus* — un sistema di dimostrazione algebraico che estende la risoluzione — usando argomenti di rango su spazi vettoriali su campi finiti. Il risultato sulle caratteristiche diverse menzionato nell'Appendice L è contenuto qui, in forma tecnica. Il metodo del rango è uno dei più potenti disponibili in proof complexity, ma si applica solo a sistemi algebraicamente strutturati.

Haken, A. (1985). The intractability of resolution. *Theoretical Computer Science*, 39(2–3), 297–308.

Il primo lower bound esponenziale per la risoluzione sulle tautologie di pigeonhole. Il risultato — $\text{Size}(\text{PHP}_n^{n+1} \vdash \bot) \geq 2^{n/20}$ — è menzionato nell'Appendice I e costituisce il riferimento storico per l'intera linea di ricerca sui lower bound per la risoluzione. La tecnica usata da Haken — misure di complessità sui sistemi di clausole — è diversa da quella di Ben-Sasson e Wigderson, e i due approcci si integrano.

Kushilevitz, E., & Nisan, N. (1997). *Communication Complexity*. Cambridge University Press.

Il riferimento standard per la complessità della comunicazione, lo strumento usato nella dimostrazione del lower bound di Krajíček. La complessità di comunicazione misura quanti bit devono scambiarsi due agenti per calcolare congiuntamente una funzione di cui ciascuno conosce solo una parte dell'input. Il collegamento con il gap

di rappresentazione del Capitolo VI è diretto: il costo di traduzione $CT(\alpha \to \beta; x)$ può essere lower-bounded dalla complessità di comunicazione del problema di "trasmettere la comprensione di x da α a β".

Gärdenfors, P. (2000). *Conceptual Spaces: The Geometry of Thought*. MIT Press.

Gärdenfors propone di modellare i concetti come regioni in spazi metrici multidimensionali — gli *spazi concettuali* — e la comunicazione come il problema di trasmettere la struttura di tali regioni. Il framework è geometrico e non computazionale, ma converge con l'analisi del gap di rappresentazione del Capitolo VI: sistemi concettuali diversi corrispondono a metriche diverse sullo stesso insieme di oggetti, e la traduzione tra sistemi diversi richiede un overhead che cresce con la dissimilarità delle metriche. La monografia non adotta il framework di Gärdenfors — che non ha la struttura formale necessaria per i risultati del Capitolo VI — ma lo riconosce come percorso parallelo che raggiunge conclusioni compatibili per via diversa.

IX. Oblio, memoria esperta e consolidamento mnestico

Ebbinghaus, H. (1885). *Über das Gedächtnis: Untersuchungen zur experimentellen Psychologie*. Duncker & Humblot. (Traduzione inglese: *Memory: A Contribution to Experimental Psychology*, 1913, Teachers College, Columbia University.)

Il punto di partenza obbligatorio per qualsiasi teoria dell'oblio. Ebbinghaus documenta la curva di dimenticanza — il decadimento esponenziale della ritenzione nel tempo in assenza di ripasso — attraverso esperimenti sistematici su sé stesso. La curva di Ebbinghaus descrive l'oblio *non selettivo* — la perdita indifferenziata di informazione — che è il caso opposto rispetto all'oblio produttivo discusso nel Capitolo VII. La monografia si distanzia da Ebbinghaus precisamente su questo punto: la curva di Ebbinghaus misura il decadimento del residuo d_x, non la dinamica del modello p_x, e le due quantità hanno traiettorie temporali radicalmente diverse.

de Groot, A. D. (1946/1978). *Thought and Choice in Chess* (2ª ed.). Mouton. (Originale olandese: *Het denken van den schaker*, 1946.)

Lo studio classico sulla cognizione scacchistica. De Groot mostrò che i maestri di scacchi non differiscono dai novizi nella velocità di analisi delle mosse o nella profondità della ricerca, ma nella capacità di *percepire* la struttura della posizione — di riconoscere configurazioni significative in modo immediato. L'esperimento del ricordo di posizioni casuali vs posizioni di partita reale (replicato sistematicamente da Chase e Simon) è la dimostrazione empirica che gli esperti hanno un modello p_x di alta sofisticazione e che questo modello non aiuta con il residuo d_x casuale. Rilevante per VII.4 e VII.5.

Chase, W. G., & Simon, H. A. (1973). Perception in chess. *Cognitive Psychology*, 4(1), 55–81.

La replica e l'estensione sistematica del lavoro di de Groot. Chase e Simon quantificano la differenza tra maestri e principianti in termini di *chunks* — unità percettive che corrispondono a configurazioni significative. Il maestro percepisce la posizione come un insieme di chunks (alta sofisticazione) mentre il principiante percepisce pezzi isolati (bassa sofisticazione). Il numero di chunks che un maestro deve memorizzare per ricostruire una posizione di partita è comparabile a quello di un principiante per posizioni casuali. Questo è il correlato empirico della conservazione di D_s(p_x) e della perdita di d_x con l'expertise.

Ericsson, K. A., Krampe, R. T., & Tesch-Römer, C. (1993). The role of deliberate practice in the acquisition of expert performance. *Psychological Review*, 100(3), 363–406.

Il testo che ha reso popolare la "regola delle 10.000 ore". Al di là della semplificazione giornalistica, il lavoro di Ericsson documenta che l'expertise in domini complessi richiede anni di pratica deliberata — pratica focalizzata sul confine della competenza corrente, non sulla ripetizione di ciò che si sa già. Nel quadro della monografia, la pratica deliberata è il meccanismo attraverso cui l'agente esegue successioni di Π_model: ogni sessione di pratica identifica i limiti del modello corrente p_x, genera nuovi dati d_{new} che li mettono in evidenza, e innesca una revisione del modello che aumenta D_s(p_{x'}) a spese dell'informazione accidentale eliminata.

Stickgold, R. (2005). Sleep-dependent memory consolidation. *Nature*, 437(7063), 1272–1278.

La rassegna più citata sul consolidamento mnestico durante il sonno. Stickgold sintetizza le evidenze che mostrano come il sonno REM e il sonno a onde lente abbiano funzioni distinte nel consolidamento: il sonno REM sembra favorire la generalizzazione e l'astrazione, il sonno a onde lente il consolidamento delle tracce specifiche. Nel quadro della monografia, il sonno REM è il momento privilegiato della proiezione Π_model: le tracce di d_x vengono soppresse mentre la struttura p_x viene consolidata e integrata con il modello globale del dominio. La proposta è interpretativa — la neuroscienze non usa il linguaggio di KC e sofisticazione — ma le evidenze comportamentali sono consistenti con essa.

Walker, M. P., & Stickgold, R. (2004). Sleep-dependent learning and memory consolidation. *Neuron*, 44(1), 121–133.

Estende e sistematizza il lavoro di Stickgold con particolare attenzione alla distinzione tra memoria dichiarativa e procedurale. La memoria procedurale — le abilità motorie e cognitive automatizzate — consolida prevalentemente durante il sonno a onde lente. Nel quadro della monografia, la memoria procedurale corrisponde al modello p_x in forma di programma eseguibile — non di enunciato memorizzato ma di procedura disponibile — e il suo consolidamento notturno è la controparte neurale del processo di proiezione al modello discusso in VII.5.

McClelland, J. L., McNaughton, B. L., & O'Reilly, R. C. (1995). Why there are complementary learning systems in the hippocampus and neocortex: Insights from

the successes and failures of connectionist models of learning and memory. *Psychological Review*, 102(3), 419–457.

Il paper che fonda il *Complementary Learning Systems* (CLS) framework. McClelland, McNaughton e O'Reilly propongono che ippocampo e neocortex realizzino due tipi di apprendimento strutturalmente distinti: l'ippocampo apprende rapidamente episodi specifici (acquisizione di d_x), il neocortex apprende lentamente strutture generali (costruzione di p_x). Il dialogo tra i due sistemi durante il sonno è il meccanismo di consolidamento. Il CLS framework è il correlato neurale più diretto della distinzione modello-residuo della monografia, e il suo sviluppo formale avrebbe richiesto la distinzione KC–D–soph che non era disponibile agli autori.

Anderson, J. R. (1983). *The Architecture of Cognition*. Harvard University Press.

Anderson propone il framework ACT* (Adaptive Control of Thought) in cui la conoscenza dichiarativa — fatti e proposizioni — viene gradualmente *proceduralizzata* in conoscenza procedurale — regole di produzione automatizzate — attraverso l'uso ripetuto. Il processo di proceduralizzazione corrisponde, nella monografia, alla trasformazione di d_x in parte di p_x: ciò che era informazione specifica da recuperare diventa primitiva computazionale incorporata nel modello. Anderson non usa il linguaggio della complessità algorítmica, ma il meccanismo che descrive è formalmente interpretabile nel quadro KC–D–soph.

X. Filosofia e storia della comprensione scientifica

Kuhn, T. S. (1962). *The Structure of Scientific Revolutions*. University of Chicago Press.

Il testo con cui qualsiasi discussione sui cambi di paradigma deve confrontarsi. Kuhn descrive la scienza normale come puzzle-solving all'interno di un paradigma — un'operazione di bassa D nel quadro della monografia — e la rivoluzione scientifica come sostituzione del paradigma. Il collegamento con il gap di rappresentazione del Capitolo VI è diretto: i paradigmi incommensurabili di Kuhn sono sistemi di rappresentazione con gap di simulazione non costante. Il termine "incommensurabilità" — che Kuhn usa in senso informale — acquista nel quadro della monografia un significato formale preciso: due paradigmi sono incommensurabili se il loro gap di simulazione cresce senza limite con la profondità degli oggetti trattati. La monografia non è kuhniana — la tesi che i paradigmi siano incommensurabili è più forte di ciò che il gap di simulazione implica — ma usa Kuhn come motivazione fenomenologica per il problema formale.

Galison, P. (1997). *Image and Logic: A Material Culture of Microphysics*. University of Chicago Press.

Galison propone il concetto di *zona di scambio* (trading zone): lo spazio in cui comunità scientifiche con culture, strumenti e sistemi di rappresentazione diversi riescono a collaborare senza condividere un linguaggio comune, attraverso la costruzione di *creoli*

— linguaggi ibridi che permettono comunicazione locale senza integrazione globale. Nel quadro della monografia, una zona di scambio è un meccanismo per ridurre CT(α → β; x) per oggetti x specifici senza ridurre il gap di simulazione globale: si costruisce un sistema di primitive condiviso per un sottoinsieme limitato di oggetti, lasciando incommensurabile il resto. La ricchezza storica e empirica del lavoro di Galison complementa la struttura formale del Capitolo VI.

Wittgenstein, L. (1953). *Philosophische Untersuchungen / Philosophical Investigations*. Blackwell.

Le *Ricerche filosofiche* sono rilevanti per il Capitolo VI in due punti specifici. Il concetto di *Sprachspiel* (gioco linguistico) come sistema di pratiche condivise in cui le parole acquistano significato è la controparte filosofica del sistema di rappresentazione M_α: un agente che non condivide il gioco linguistico del dominio non ha accesso alle primitive necessarie per decodificare i messaggi di chi vi partecipa. Il concetto di *seguire una regola* e la sua irriducibilità a qualsiasi formulazione esplicita finita è la controparte filosofica dell'indecidibilità di KC: non si può specificare completamente, in un sistema formale finito, cosa significa seguire correttamente una regola. La monografia non adotta la posizione antiteorica di Wittgenstein — che ritiene impossibile qualsiasi formalizzazione adeguata — ma riconosce che i suoi argomenti delimitano il dominio di applicazione del quadro formale.